UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS

Daniel Martins Lima

SISTEMA EMBARCADO DE CONTROLE PREDITIVO PARA PROCESSOS INDUSTRIAIS

Florianópolis

2013

Daniel Martins Lima

SISTEMA EMBARCADO DE CONTROLE PREDITIVO PARA PROCESSOS INDUSTRIAIS

Dissertação de mestrado submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Mestre em Engenharia de Automação e Sistemas.
Orientador: Prof. Dr. Julio Elias Normey-Rico
Coorientador: Prof. Dr. Guilherme Vianna Raffo

Florianópolis

2013

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Lima, Daniel Martins
Sistema Embarcado de Controle Preditivo para Processos Industriais / Daniel Martins Lima ; orientador, Julio Elias Normey-Rico ; co-orientador, Guilherme Vianna Raffo. - Florianópolis, SC, 2013.
171 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Inclui referências

1. Engenharia de Automação e Sistemas. 2. Controle de Processos. 3. Controle Preditivo. 4. Sistema Embarcado. I. Normey-Rico, Julio Elias. II. Raffo, Guilherme Vianna. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Daniel Martins Lima

SISTEMA EMBARCADO DE CONTROLE PREDITIVO PARA PROCESSOS INDUSTRIAIS

Esta Dissertação de mestrado foi julgada aprovada para a obtenção do Título de **"Mestre em Engenharia de Automação e Sistemas"**, e aprovada em sua forma final pelo **Programa de Pós-Graduação em Engenharia de Automação e Sistemas**.

Florianópolis, 20 de março de 2013.

Prof. Dr. Jomi Fred Hübner
Coordenador do Curso

Banca Examinadora:

Julio Elias Normey-Rico
Presidente – Orientador – DAS – UFSC

Alexandre Sanfelice Bazanella – DEE - UFRGS

Antônio Augusto Rodrigues Coelho – DAS – UFSC

Leandro Buss Becker – DAS – UFSC

Agradecimentos à minha família que me permitiu chegar até aqui.

AGRADECIMENTOS

Ao professor Julio Elias Normey-Rico pela concepção e orientação deste trabalho.

Ao Centro Nacional de Desenvolvimento Científico e Tecnológico pela concessão da bolsa de mestrado.

À empresa ATTA pela parceria no projeto e permitir a realização deste trabalho.

Ao graduando em Engenharia de Controle e Automação Rafael Sartori pelo auxílio em vários dos experimentos.

Aos amigos e família pelo apoio.

RESUMO

Controle Preditivo Baseado em Modelo (MPC) é uma metodologia de controle comumente utilizada nos setores petroquímicos e químicos da indústria para o controle de grandes processos. Seu sucesso é devido ao fato possibilitar o desenvolvimento de algoritmos de controle multivariável com compensação intrínseca de atraso de transporte, também é capaz de lidar com as restrições do processo e seus parâmetros de ajuste interpretados no domínio do tempo são intuitivos. No entanto, seu uso na indústria é limitado a grandes processos onde o investimento em sistemas avançados de controle é economicamente viável, devido aos altos custos das soluções vendidas atualmente. Este trabalho propõe um sistema embarcado com controle preditivo que possa ser adquirido por companhias que tenham processos de pequeno e médio porte que poderiam obter ganhos de produtividade com o uso do MPC. Este documento vai descrever todos os passos realizados para desenvolver tal equipamento: o estudo da tecnologia usada no controle de processos atualmente e de vários algoritmos MPC, a metodologia de engenharia de *software* utilizada para desenvolver um programa bem estruturado e, finalmente, os experimentos feitos para validar o protótipo criado. Estes resultados incluem uma série de simulações *Hardware-in-the-loop* e um teste de integração com um processo real.

Palavras-chave: Controle de Processos. Controle Preditivo. Sistemas Embarcados.

ABSTRACT

Model Predictive Control is a control methodology commonly used in petrochemical and chemical fields to control large processes. Its success can be traced to the facts that it can deal with MIMO processes, it has intrinsic dead-time compensation, it is capable to handle the process' constraints and its tuning, interpreted in the time domain, are easily understandable. Nevertheless, its usage in the industry is limited to large processes where the investment in the currently expensive advanced control systems are economically viable. This work proposes a low cost embedded MPC controller targeted at companies with medium and small processes that could have production improvements with the use of MPC. This document will describe all the steps taken to develop such equipment: the study of current process' control technologies and of various MPC algorithms, the software engineering methodology adopted to obtain a well structured software and, finally, the experiments done to validate the prototype. These results include a series of Hardware-in-the-loop simulations and an integration test with a real process.

Keywords: Process Control. Model Predictive Control. Embedded Systems.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CARIMA	*Controller Auto-Regressive Integrated Moving-Average*
CID	*Controller Interface Device* ou, em português, Dispositivo Controlador de Interface
CLP	Controlador Lógico Programável
CVXOPT	*Convex Optimization Package*
DDC	*Direct Digital Control*
DCS	*Distributed Control System*
DMF	Descrição Matricial Fracionária
DMC	*Dynamic Matrix Control*
DTCGPC	*Dead-Time Compensation Generalized Predictive Controller*
E/S	Entrada/Saída
FCS	*Field Control Systems*
FF	FOUNDATION Fieldbus
FPGA	*Field Programmable Gate Array*
GPC	*Generalized Predictive Control*
HIL	*Hardware-in-the-Loop*
ISC	*Integral Square Controller Effort* ou, em português, Esforço de Controle Quadrático Integral
ISE	*Integral Square Error* ou, em português, Erro Quadrático Integral
LAS	*Link Active Scheduler*
MFD	*Matrix Fraction Description*
MIMO	*Multiple-Inputs Multiple-Output*
MISO	*Multiple-Inputs Single-Output*
PIS	Programa de Interface da Simulação
PNMPC	*Practical Non-linear Model Predictive Control*
POO	Projeto Orientado a Objetos
PSF	Preditor de Smith Filtrado
SISO	*Single Input Single Output*
SO	Sistema Operacional
SOTR	Sistemas Operacionais de Tempo Real
SSMPC	*State Space Model Predictive Control*

UML	*Unified Modeling Language*
UP	*Unified Process* ou, em português, Processo Unificado
XML	*Extensible Markup Language*

LISTA DE SÍMBOLOS

$\boldsymbol{x}$	Variáveis minúsculas em negrito representam vetores
$\boldsymbol{X}$	Variáveis maiúsculas em negrito representam matrizes
$X(z^{-1})$	Variáveis maiúsculas representam polinômios
$\boldsymbol{X}(z^{-1})$	Representa uma matriz cujos elementos são polinômios
Δ	Representa o operador diferença $\Delta = 1 - z^{-1}$
m	Quantidade de entradas de um sistema MIMO
n	Quantidade de saídas de um sistema MIMO
p	Quantidade de perturbações de um sistema MIMO
N_{1i}	Horizonte inicial de predição da i-ésima saída
N_{2i}	Horizonte final de predição da i-ésima saída
N_i	Tamanho do horizonte de predição da i-ésima saída $N_i = N_{2i} - N_{1i} + 1$
N_{ui}	Horizonte de controle da i-ésima entrada
$\boldsymbol{\lambda}$	Matriz diagonal de ponderação do incremento de controle onde o i-ésimo elemento da diagonal é a ponderação no tempo $t + i - 1$
$\boldsymbol{\delta}$	Matriz diagonal de ponderação dos erros futuros onde o i-ésimo elemento da diagonal é a ponderação no tempo $t + N_1 + i - 1$
$\boldsymbol{Q_y}$	Ponderação dos erros futuros de um sistema MIMO: $\boldsymbol{Q_y} = diag(\boldsymbol{\delta_1}; \ldots; \boldsymbol{\delta_n})$
$\boldsymbol{Q_u}$	Ponderação dos incrementos futuros de controle de um sistema MIMO: $\boldsymbol{Q_u} = diag(\boldsymbol{\delta_1}, \ldots, \boldsymbol{\delta_n})$
$\boldsymbol{w}$	Vetor de referências futuras
$\hat{\boldsymbol{y}}$	Vetor de predições ótimas do processo
$\boldsymbol{u}$	Vetor com os incrementos futuros de controle
f	Vetor com a resposta livre do sistema
$\boldsymbol{H}$	Matriz que multiplica o vetor de incrementos futuros de controle
$\boldsymbol{F}$ ou $\boldsymbol{F}(z^{-1})$	Matriz que multiplica os valores passados das saídas do processo
$\boldsymbol{H_v}$ e $\boldsymbol{I_v}$	Matrizes que multiplicam, respectivamente, os valores

	futuros e passados das perturbações
$\boldsymbol{I}$	Representa, dependendo do contexto, uma matriz identidade ou a matriz que multiplica os incrementos passados de controle
$len(\cdot)$	Função que retorna o número de coeficientes de um polinômio.
$diag(\cdot)$	Representa uma matriz quadrada diagonal cujos elementos são dados pelos parâmetros dados
$\boldsymbol{P}, \boldsymbol{q}, f_o$	Variáveis utilizadas para representar o problema de otimização quadrático
$\overline{\boldsymbol{R}}$, $\overline{\boldsymbol{c}}$	Matrizes que representam as restrições do problema quadrático resolvido pelos algoritmos MPC
$\mathbf{1}_x$	Vetor de dimensão x cujos elementos são todos iguais a 1
T_s	Período de amostragem

SUMÁRIO

1 INTRODUÇÃO

Na indústria de processos os controladores preditivos (MPC - *Model based Predictive Controllers*) são largamente utilizados no setor petroquímico, principalmente nas refinarias. A formulação MPC integra controle ótimo, controle estocástico, compensação de atraso de transporte de processos, controle multivariável, controle *feed-forward* de perturbações mensuráveis e também é capaz de considerar referências futuras [5].

O MPC não é uma estratégia de controle específica, é o nome dado a um conjunto de métodos de controle que foram desenvolvidos considerando o conceito de predição e a obtenção do sinal de controle através da minimização de uma determinada função objetivo [5]. Esta função considera o erro futuro, calculado a partir das referências futuras e das saídas do modelo, o esforço de controle, além das restrições nas variáveis de processo e/ou de controle que possam existir.

Diversas pesquisas têm mostrado que a utilização destes controladores permite melhorar a qualidade da produção industrial [6]. Porém, na indústria, o uso destes controladores está restrito a processos de grande porte onde o investimento em sistemas avançados de controle é viável economicamente. Nestes processos, o MPC é utilizado em cascata com controladores clássicos PID [7] que controlam as variáveis fundamentais do processo: vazão, temperatura, nível, etc., isto é, os sinais de controle gerados pelo MPC são as referências para os controladores locais.

Apesar do sucesso obtido com o MPC em vários setores da indústria, principalmente nas refinarias de petróleo, o uso deste tipo de algoritmo não é muito difundido em outras partes do processo, como por exemplo, no processo de produção e exploração, nem em plantas de menor porte, onde poderia trazer ganhos significativos.

Alguns dos motivos da pouca disseminação do MPC nestes setores são: o custo do sistema de controle; a falta de sistemas de menor porte de baixo custo; a dificuldade de adaptar estes controladores a plantas existentes.

Para ilustrar alguns desses motivos, pode-se citar, por exemplo, a solução MPC vendida pela empresa *Emerson Process Management*, o pacote de *software* DeltaV Predict. Este pacote implementa o algoritmo DMC (*Dynamic Matrix Control*) de controle preditivo e é capaz de lidar com até 80 variáveis manipuladas e 40 variáveis controladas. Além disso, possui algoritmos de identificação de modelo para agili-

zar a configuração e ajuste do controlador [8]. Mas para utilizar este *software*, é necessário ter também o programa DeltaV v12.3 que roda em uma estação de trabalho com um *hardware* específico vendido pela própria empresa. Também é requerido o uso de um controlador série-M, produzido pela *Emerson*, que faz a comunicação com os dispositivos do processo e a estação de trabalho. O custo destes programas e equipamentos chega a algumas dezenas de milhares de dólares, o que não inclui custos de instalação e integração de equipamentos, e suporte técnico.

Existem muitos trabalhos na área de controle preditivo que tentam resolver todos ou pelo menos alguns dos problemas listados. Uma linha de pesquisa é a programação de um *Field Programmable Gate Array* (FPGA) de forma a otimizar a execução de algoritmos MPC [9–11]. FPGA é um tipo de circuito integrado que pode ser configurado de acordo com a aplicação do usuário e estes trabalhos exploram algumas peculiaridades dos algoritmos MPC para aumentar a velocidade do procedimento de cálculo da ação de controle. Outro tipo de solução foi proposta em [12], onde um algoritmo MPC é embarcado em um controlador lógico programável (CLP) usando o padrão de programação IEC 61131-3. Estes projetos propõem soluções que tentam disseminar o uso de algoritmos preditivos com equipamentos de baixo nível, por exemplo, atuadores, desta forma, eles conseguem obter tempos de amostragem na ordem de milissegundos mesmo considerando restrições. Apesar disto, o número de restrições e de variáveis de controle são muito limitados devido às limitações dos equipamentos sendo utilizados.

Assim, este trabalho, desenvolvido em parceria com a empresa ATTA [13], apresenta uma solução diferente para resolver os problemas listados anteriormente. Planeja-se a criação de um sistema embarcado de baixo custo com componentes disponíveis a pronta-entrega que possuem maiores recursos (memória, velocidade de processador, etc.), se comparado com um FPGA ou CLP, de forma a permitir o controle de pequenos e médios processos com dinâmicas lentas do mesmo modo como são utilizados os pacotes profissionais de MPC, isto é, fornecendo as referências para os controladores de baixo nível que controlam as variáveis básicas do processo. Desta forma, estes tipos de planta poderão ter acesso a esta tecnologia de controle ótimo multivariável que, quando bem ajustado, permite a melhoria da qualidade da produção [6]. O protótipo a ser criado deve possuir as seguintes características:

- fácil integração com processos industriais existentes através do suporte a diferentes redes industriais, por exemplo, Fieldbus, MODBUS, PROFIBUS-PA;

- configuração rápida dos parâmetros do controlador e do próprio dispositivo através de uma interface de usuário;
- suporte a diferentes algoritmos MPC, permitindo que o usuário escolha a melhor opção para um determinado processo;
- baixo custo.

O capítulo 2 deste documento fará uma breve descrição da história do controle de processos e da tecnologia utilizadas nos dias atuais. No capítulo 3 será feita uma descrição do *hardware* do protótipo do sistema embarcado, assim como todo o conjunto de *softwares* utilizado para o desenvolvimento do programa que fará o controle de processos. A revisão dos algoritmos MPC implementados será feita no capítulo 4. No capítulo 5 será abordada a metodologia de desenvolvimento de *software* utilizada para desenvolver os programas deste projeto. No capítulo 6 serão apresentadas os resultados obtidos e, para finalizar, as conclusões estarão presentes no capítulo 7.

2 CONTROLE DE PROCESSOS

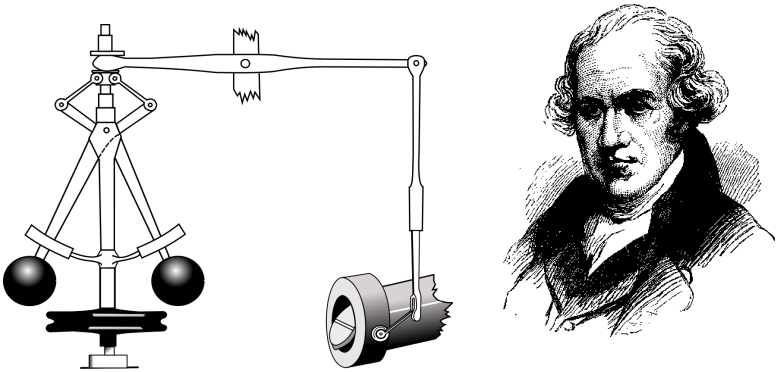
Nos tempos atuais, percebe-se a crescente presença de sistemas automáticos de controle. Máquinas ocupam cada vez mais espaço dentro das residência e indústrias. Toda esta revolução que vem acontecendo na sociedade contemporânea não passa do desabrochar de uma tecnologia que já é estudada há tempos e continua sendo muito importante: o controle de sistemas [14].

Grandes avanços no controle de processos foram obtidos através da história, por exemplo, é sabido que os Romanos utilizaram sistemas de válvulas engenhosos em seus aquedutos para manter o nível da água constante. Alguns especialistas afirmam que, na Mesopotâmia antiga, mais de 2000 anos A.C., o controle do sistema de irrigação também era uma arte conhecida [15]. Mas foi com o advento do governador centrífugo desenvolvido por James Watt, na Fig. (1a), que houve um avanço considerável nesta área. O governador de Watt, composto basicamente de um par de esferas de metal montadas em um dispositivo mecânico ligado a uma válvula de escape, era utilizado para controlar a velocidade de rotação de motores a vapor. Neste equipamento, quando a velocidade do motor aumentava demasiadamente, a força centrífuga fazia o raio de rotação das esferas aumentar, movimentando o dispositivo mecânico que fazia a válvula de escape abrir, diminuindo a pressão interna do motor, o que, por sua vez, diminuía a velocidade de rotação [16].

Equipamentos como o governador centrífugo, apesar de proporcionarem grandes avanços na indústria na época, não constituíam sistemas distribuídos de controle pois operavam localmente e independentemente de qualquer outro equipamento que o processo pudesse ter. Apenas com a utilização de transmissão de sinais é que foi possível haver a composição de sistemas distribuídos [17].

2.1 SISTEMAS DISTRIBUÍDOS DE CONTROLE DE PROCESSOS

Sistemas distribuídos de controle de processos são utilizados na indústria de forma a permitir a otimização da produtividade, aumentar a qualidade da produção, diminuir custos, melhorar a segurança, entre outros. Sistemas são uma combinação de componentes que atuam em conjunto e realizam um certo objetivo [14]. Na área de controle, há a composição de um sistema distribuído quando os equipamentos que



(a) (b)

Figura 1 – Esquematização do governador de Watt em (a) [1], e James Watt (1736-1819) em (b)

permitem o controle de um processo são interligados entre si através de uma rede. A rede permite o acesso remoto a estes equipamentos, permitindo algoritmos de controle mais eficientes e uma segurança maior. Foi a criação das redes industriais que permitiu a existência destes sistemas, e elas só vieram a existir devido à evolução na área de transmissão de sinais. Na verdade, cada aperfeiçoamento feito nesta área acabou por conduzir a uma evolução das redes e, por consequência, dos sistemas de controle, por possibilitarem um melhor acesso ao processo, seja pela leitura de dados através dos sensores ou pelo envio de dados aos equipamentos de atuação [17].

Um passo fundamental na evolução da transmissão de sinais foi o advento da transmissão analógica, que permitia que os dados de entrada e saída (E/S) do processo fossem enviados ou recebidos remotamente por uma central de controle através de sinais elétricos analógicos de corrente. Isto permitiu, nas décadas de 1950 a 1970, o desenvolvimento do chamado *Direct Digital Control* (DDC), uma arquitetura centralizada, composta de um ou mais controladores responsáveis pelo processo que eram conectados aos equipamentos de E/S. Estes controladores eram configurados localmente, pois só se transmitia pela rede dados de E/S do processo, assim, possuíam painéis localizados na sala de controle por onde eram feitas as configurações. Como os controladores precisavam estar localizados em salas especiais, e cada equipamento transmitia seus dados por um par de cabos, a instalação e manutenção

Figura 2 – Computador Argus Ferranti de 1961, utilizado em arquiteturas DDC [2]

destes sistemas era extremamente trabalhosa e custosa. Além disso, caso houvesse uma falha nos computadores centrais do sistema, o que não era incomum, haveria uma falha generalizada do processo, fazendo com que fosse necessária a existência de equipamentos servomecânicos e/ou pneumáticos em espera para que não houvesse uma paralisação duradoura da produção [17]. Na Fig. (2) é mostrado um computador central utilizado em DDCs.

Ao longo da década de 1970 houve o desenvolvimento da comunicação digital. O advento desta tecnologia é um grande marco na indústria pois permitiu que múltiplos dados pudessem ser transmitidos em um par de cabos, o que não acontecia com a comunicação analógica, e, por admitirem somente dois estados (um e zero), os sinais digitais são mais robustos que os analógicos, sendo menos suscetíveis a distorções causadas por ruído ou interferências elétricas. Além disso, a comunicação digital pode utilizar técnicas de checagem de erro para detectar uma distorção e retransmitir o sinal, o que não é possível no caso analógico [3]. Pode-se dizer mais, a comunicação digital permitiu o uso de um mesmo barramento de comunicação (ou cabo de transmissão) por múltiplos equipamentos, reduzindo ainda mais a infra-

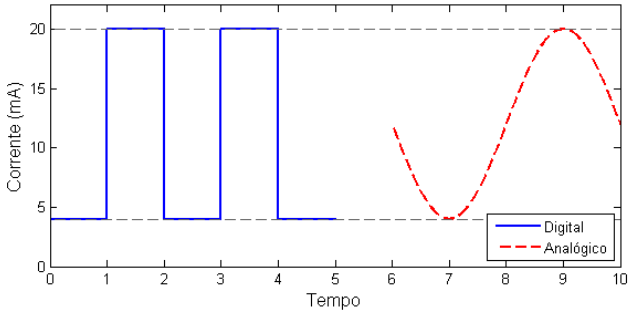


Figura 3 – Comparação entre dois sinais, um sinal analógico (direita) e digital (esquerda), transmitidos por corrente (4-20mA).

estrutura necessária para a instalação de equipamentos. A Fig. (3) mostra uma comparação entre os sinais analógicos e digitais transmitidos por corrente (4-20 mA). No caso digital, idealmente, os estados 0 e 1 são representados pelos valores de corrente 4 mA e 20 mA, respectivamente. Caso seja lido um valor intermediário, este é considerado inválido. Para montar uma mensagem, são enviados sequências de 0s e 1s com tamanho pré-determinado de elementos. Essa mensagem é então interpretada e pode conter dados de múltiplos equipamentos. No lado direito da mesma figura está um exemplo de sinal analógico, que pode variar continuamente entre os limites de 4 mA e 20 mA. Dado um sensor analógico de temperatura que pode medir entre 0 e 100 ^oC, por exemplo, pode-se fazer com que a leitura de 0 ^oC seja equivalente a 4 mA e que 100 ^oC seja equivalente a 20 mA, havendo uma equivalência linear entre os valores intermediários. Então, nos casos analógicos, o sinal só pode representar uma variável, neste exemplo, o valor de temperatura.

Como dito anteriormente, cada evolução na transmissão de sinais implicou numa evolução no controle de processos. Neste caso, um dos resultados foi a arquitetura Sistema Distribuído de Controle ou, em inglês, *Distributed Control System* (DCS). Estes sistemas são ditos distribuídos pois, ao contrários dos DDCs, os processos são divididos em sub-sistemas que são controlados, por exemplo, pelos chamados Controladores Lógico Programáveis (CLPs).

A subdivisão do processo em sub-sistemas traz diversas vantagens. A principal delas é que, como o controle do processo está dividido em sistemas menores (os CLPs), caso haja algum erro em um deles, o

risco de causar uma falha geral no processo é menor. Além disso, se houver redundância de controladores, ou seja, a existência de CLPs de reserva que entram em ação caso o controlador principal do sub-sistema falhe, este risco de falha é reduzido ainda mais, mas isto leva, é claro, a custos maiores [17].

Apesar do nome distribuído, os DCSs, pelos padrões atuais, são considerados sistemas centralizados [17]. O avanço nas tecnologias de redes industriais em conjunto com a evolução tecnológica de processadores, que reduziu custos e miniaturizou equipamentos eletrônicos, permitiu que fossem criados dispositivos inteligentes capazes de auto-regulação. Por exemplo, o Posicionador de Válvula Fieldbus Foundation [18], é utilizado para regulação da abertura de uma válvula. O controle de abertura é feito por este mesmo dispositivo, bastando apenas configurar o algoritmo de controle remotamente e enviar a referência de abertura desejada, tudo realizado através da rede industrial.

O desenvolvimento destes tipos de dispositivos levou à arquitetura Sistema de Controle de Campo ou, em inglês, *Field Control Systems* (FCS), onde há a distribuição da habilidade de controle aos instrumentos de campo, como o exemplo citado anteriormente. O FCS leva o DCS a um passo adiante aumentando a tolerância a falhas, pois, como cada dispositivo não lida com mais do que uma malha de controle, o problema de uma única falha afetar uma grande parte da planta é quase eliminado [17]. O avanço provido por estes dispositivos inteligentes vai muito mais além, já que estes são capazes de se diagnosticar e reportar qualquer problema através da rede, reduzindo o tempo necessário para a descoberta do local da falha e sua reparação. A Fig. (4) faz uma comparação entre as arquiteturas descritas, indicando onde são executados os algoritmos de controle em cada uma.

Atualmente, estamos em um período de transição da arquitetura DCS para FCS. Novas instalações tendem a ser construídas com dispositivos inteligentes utilizados nos FCSs, mas ainda existem inúmeras instalações mais antigas funcionando com a estrutura DCS que não podem custear arquiteturas mais novas [17]. Assim, para este projeto, seria de grande interesse criar um dispositivo inteligente capaz de ser utilizado tanto em arquiteturas DCS como FCS.

As seções seguintes tratarão das estruturas DCS e FCS de modo a ficar claro como o dispositivo a ser desenvolvido será inserido em cada uma destas arquiteturas de controle.

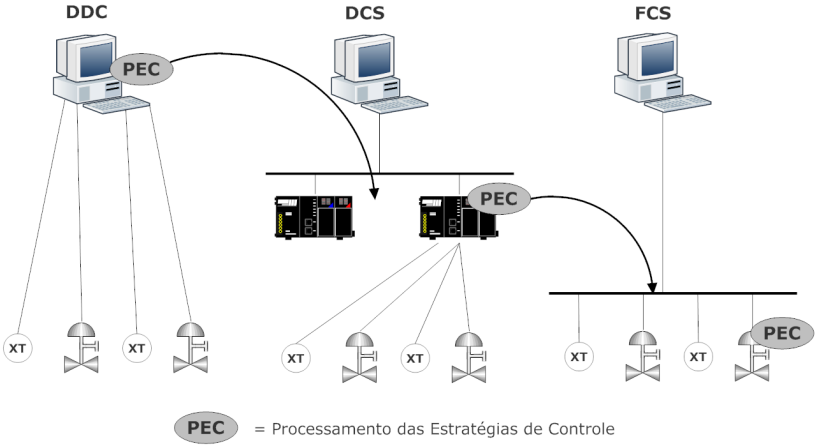


Figura 4 – Comparação entre as arquiteturas DDC, DCS e FCS [3]

2.2 ARQUITETURA DCS

A Fig. (5) mostra uma esquematização de um sistema de controle do tipo DCS. Este tipo de arquitetura pode ter até quatro níveis de rede, cada uma funcionando com uma tecnologia diferente [17]. O nível mais baixo é o dos dispositivos (sensores, atuadores, etc.). Como muitos dos instrumentos ainda utilizam comunicação analógica por corrente, é necessário o uso de sistemas de aquisição de dados que convertem a informação analógica em digital. O segundo nível é o de E/S Remoto, que faz a conexão entre os sistemas de aquisição com os controladores locais, que são geralmente CLPs, mas podem ser outros tipos de computadores industriais. Estes controladores locais são os responsáveis por executar os algoritmos de controle, lógicas de intertravamento e, em alguns casos, também servir de *gateway* para os níveis mais baixos do processo.

O terceiro nível é o de controle, que interliga todos os controladores locais e assim permite o acesso a todo o processo por terminais espalhados pela fábrica. Por último, o nível de processo, que conecta processos distintos aos computadores que fazem a gerência da produção local. As redes do segundo e terceiro níveis são, geralmente, redes industriais, que são redes com propriedades especiais, tal como poder operar em ambientes com interferências eletromagnéticas. Existe um grande número desse tipo de redes com características diferentes. Por exemplo,

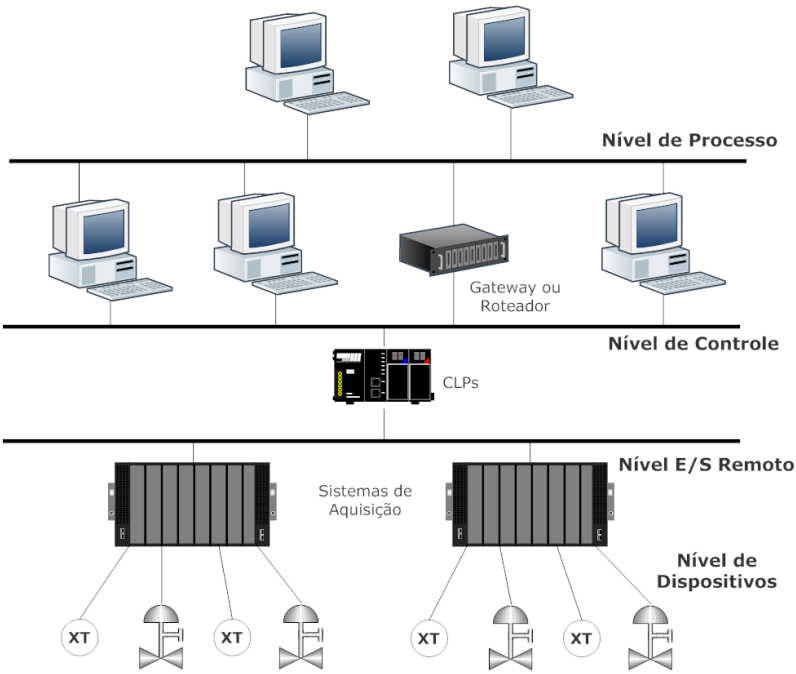


Figura 5 – Esquematização dos diferentes níveis de redes presentes em uma arquitetura DCS

para processos onde os dados são em sua maioria discretos, como em uma linha de produção automotiva onde os equipamentos são chaves de nível, botoeiras, sinaleiras luminosas, existem as redes SERIPLEX, INTERBUS-S e AS-I. Para processos contínuos, tais como os existentes em mineradoras e refinarias, utiliza-se PROFIBUS-PA, HART, MODBUS, entre outras. Cada uma destas redes possui vantagens e desvantagens que devem ser estudadas ao se instrumentar um processo, pois muitos destes padrões são proprietários e não compatíveis uns com os outros, assim, a escolha de uma rede limita bastante os equipamentos que poderão ser adquiridos.

2.3 ARQUITETURA FCS

Na arquitetura DDC, todo o sistema era centralizado em um computador. Na DCS, ocorre uma descentralização da arquitetura,

com a criação de diversos subsistemas, no entanto, o processamento dos dados ainda continua centralizado em cada subsistema (os CLPs). Na arquitetura FCS, o sistema é totalmente distribuído e o processamento das estratégias de controle é feito localmente, nos próprios instrumentos.

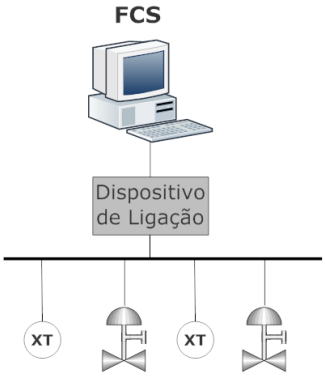


Figura 6 – Esquematização da arquitetura FCS

Hoje, a arquitetura FCS está representada basicamente pela especificação FOUNDATION Fieldbus (FF) da organização Fieldbus FOUNDATION. Ela difere dos outros protocolos de comunicação porque é orientada a resolver aplicações de controle de processos ao invés de apenas transferir dados na rede de uma forma digital. Isto se deve à inserção de processadores em cada um dos instrumentos de campo que compõem o sistema [19]. Com dispositivos inteligentes, foi possível migrar as estratégias de controle para o elemento de campo, representado pelos transmissores de temperatura, pressão, vazão e outros, e pelos atuadores, em sua maior parte válvulas de controle. Isto permite que dois ou mais instrumentos estabeleçam malhas de controle, operando de forma completamente independente [3]. Outra vantagem do FF é que, com apenas um barramento, é possível realizar a comunicação e a alimentação dos dispositivos conectados, o que reduz consideravelmente a quantidade de cabeamento necessário.

A Fig. (6) mostra uma esquematização da arquitetura FCS. Percebe-se que o FCS é mais simples do que o DCS, havendo apenas um dispositivo de ligação ou, em inglês, *Link Active Scheduler* (LAS) que realiza a função de *gateway* entre a rede FF dos dispositivos e a rede dos equipamentos de supervisão e configuração. O LAS também tem a função de coordenar a rede FF dos dispositivos, mas não realiza

o controle, que é feito pelos próprios dispositivos.

2.4 INTEGRAÇÃO ENTRE O SISTEMA EMBARCADO E REDES INDUSTRIAIS

Como dito anteriormente, seria interessante desenvolver um sistema de controle embarcado capaz de ser utilizado tanto em sistemas de arquitetura DCS quanto em FCS, pois desta forma, haveria uma clientela em potencial maior para o produto. No entanto, para que isso ocorra, o sistema embarcado deve estar apto a trabalhar com as redes industriais utilizadas em ambas as arquiteturas.

Para que um determinado equipamento consiga trabalhar com um determinado tipo de rede é preciso obedecer as normas que a definem. Por exemplo, no caso da rede FF, o meio físico e a realização da transmissão de dados são definidos pelas normas IEC 61158-2 e ISA S50.02-1992 [20]. Há também normas [21] que definem como o *software* do equipamento sendo desenvolvido deve utilizar o meio físico para interagir com outros dispositivos na rede. Como o padrão FF possui licença, apesar de ser aberto, é preciso pagar para se obter as normas. Além disso, para que o equipamento possa ser comercializado como sendo compatível com a rede FF, é preciso submeter o equipamento a uma série de testes na organização Fieldbus FOUNDATION que comprovarão a obediência às normas especificadas. Caso o equipamento passe no teste anterior, será emitido um Certificado de Registro que permitirá que o produto possa utilizar a marca da organização. Tudo isto requer o pagamento de taxas, que chegam a alguns milhares de dólares [22] e, caso seja necessário serem feitas alterações no equipamento, todo o processo deve ser repetido.

Todo o procedimento explicado anteriormente servirá apenas para a compatibilidade com um tipo de rede industrial. A rede PROFIBUS-PA, por exemplo, utilizada extensivamente em arquiteturas DCS, exige um tipo diferente de meio físico da rede FF, *software* diferente e, como também é licenciada, requer o teste e registro do equipamento.

Fica claro que, para um sistema de controle de baixo custo, este tipo de solução, de adaptar o equipamento às novas tecnologias de rede industrial, mesmo com todas as vantagens que se possa ganhar, poderia tornar o produto final inviável comercialmente, tanto pelo aumento do custo quanto pelo tempo de desenvolvimento adicional necessário.

Com isto em mente, chegou-se a uma solução: o uso do protocolo MODBUS.

2.5 MODBUS

MODBUS é um protocolo de transmissão de dados desenvolvido pela empresa *Gould Modicon*, hoje *Schneider Electric*, para o controle de processos. Em 2004 a empresa transferiu os direitos do protocolo para a *Modbus Organization* e a sua utilização é livre de taxas de licenciamento. É sabido que especificar MODBUS como a interface de um dispositivo é um meio de atingir integração de sistemas de redes diferentes, aumentando as opções de vendas e reduzindo custos. Este protocolo, devido à sua extensa utilização industrial, e pelo fato de ser aberto e livre de licenças, se tornou o padrão *de facto* quando se pensa em integração de múltiplos sistemas [20].

Este protocolo não define um meio físico para a comunicação, assim, ele pode ser implementado em vários tipos de interfaces, tais como RS232, RS485 e Ethernet. Apesar disso, certas características do MODBUS são fixas, como será visto a seguir.

2.5.1 Meio Físico

No chão de fábrica, para interligar equipamentos de campo, o meio físico mais comum quando se utiliza MODBUS é a transmissão serial RS485, ou EIA/TIA 485. Este padrão é muito útil para sistemas onde muitos instrumentos ou controladores precisam ser conectados na mesma linha. Nestas situações, um cuidado especial com o *software* deve ser tomado para coordenar qual dispositivo na rede poderá transmitir num dado momento [20]. Quando utilizado em conjunto como MODBUS, o modo de comunicação é o Mestre-Escravo, ou seja, num dado barramento, somente um dispositivo, chamado de mestre, inicia a comunicação. Desta forma, os dispositivos escravos estão sempre em espera para mensagens do mestre. Em cada mensagem há um identificador, como será visto mais adiante, que indica a qual dispositivo a mensagem é dirigida.

A interface RS485 utiliza um barramento composto de um par de cabos e a transmissão de dados é diferencial por tensão. Ela permite comunicação confiável a uma distância de 1200 m (até 5 Km a 1200 Bps), velocidades de transmissão de até 10 Mbps, além ser capaz de dividir o mesmo barramento com até 32 equipamentos, apesar de somente um poder transmitir por vez. Além disso é barata e de fácil utilização [20].

A transmissão diferencial permite uma robustez a ruídos e fun-

ciona da seguinte maneira: quando se quer transmitir o estado um, por exemplo, um dos fios do barramento assume a tensão de +5 Volts e o outro 0 Volts. Quando se quer transmitir o estado zero, a tensão nas linhas são trocadas, a que estava em +5 Volts vai para 0 e a que estava em 0 vai para +5 Volts.

2.5.2 Protocolo

As normas que definem o MODBUS especificam uma formatação da mensagem enviada, como mostrada na Tab. (1). O primeiro campo de cada mensagem, composto de 1 byte, é o de endereço (*Address Field*), que pode assumir valores de 1 a 247. Quando a mensagem é do tipo de requisição, ou seja, enviada por um dispositivo mestre, este campo assume o valor do endereço do dispositivo que deve receber a mensagem. Quando a mensagem é a resposta do dispositivo escravo, este campo assume o valor do dispositivo emissor.

Tabela 1 – Formatação da mensagem MODBUS com indicação dos tamanhos, em bytes, dos vários campos que a compõem.

Address Field	Function Field	Data Field	Error Check Field
1 byte	1 byte	Variável	2 bytes

O segundo campo (*Function Field*), de 1 byte, é o que especifica a função a ser executada pelo dispositivo que recebe a mensagem. Se o dispositivo receptor for capaz de executar a dada função, será enviada uma mensagem de resposta com o resultado da operação. As funções mais comuns são de leitura e escrita de dados, mas existem também funções de diagnóstico. Uma lista de todas as funções suportadas está na especificação do MODBUS em [23].

O terceiro campo é o de dados (*Data Field*), que tem tamanho variável de acordo com a função especificada no campo anterior. Por exemplo, quando o mestre faz uma requisição de leitura para um dispositivo, o campo de dados conterá o endereço da variável que quer ler. Neste caso, a mensagem de resposta conterá o valor da variável desejada no campo de dados.

Os últimos dois bytes da mensagem compõem o campo de checagem de erros (*Error Check Field*). O valor numérico deste campo é calculado ao se fazer o *Cyclic Redundancy Check* (CRC-16) na mensagem. Esta checagem garante que os dispositivos não receberão mensagens distorcidas durante a transmissão.

Tabela 2 – Tipos de dados especificados para o MODBUS.

Nome	Tipo	Acesso
Discrete Inputs	1 bit	Leitura
Coils	1 bit	Leitura/Escrita
Input Registers	2 bytes	Leitura
Holding Registers	2 bytes	Leitura/Escrita

As principais funções utilizadas no MODBUS (de leitura e escrita de informações) precisam ter acesso aos dados nos dispositivos. O modelo de dados MODBUS define quatro tipos, de acordo com a Tab. (2), *Discrete Input*, *Coils*, *Input Registers* e *Holding Registers*. Os dois primeiros são dados discretos (zero ou um) e os restantes são dados representados por palavras de 16 bits (2 bytes), ou seja, podem representar um valor inteiro de 0 a 65535.

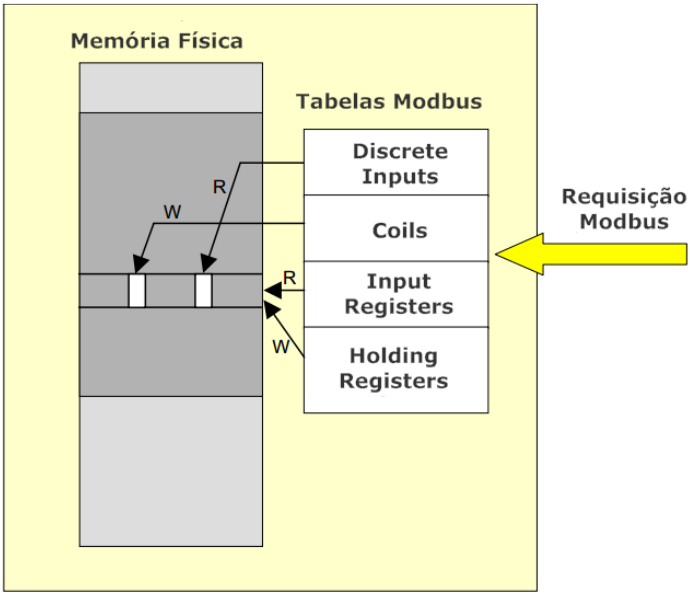


Figura 7 – Modelo de dados MODBUS e sua relação com a memória física.

Cada tipo de dado possui uma tabela própria onde são armazenadas as informações, sendo que cada elemento destas tabelas tem

um endereço fixo, que varia entre 0 e 65535, desta forma cada tabela permite a armazenagem de 65536 itens. Este endereço não deve ser confundido com o endereço em memória física. O modo como são armazenados os dados em memória depende da aplicação, desta forma é possível ter endereços diferentes no MODBUS para diferentes tipos de dados, mas que referenciam uma mesma variável na memória física. Por exemplo, é possível acessar um bit individual de uma palavra de 16 bits através da utilização de registradores *Coils* do MODBUS e também acessar a palavra completa através de um *Holding Register* com endereço diferente, como mostra a Fig. (7).

2.6 CONCLUSÕES

Este capítulo discorreu brevemente sobre a história do controle e das arquiteturas de controle distribuídos que existem hoje na indústria. Nas arquiteturas DCS e FCS, há uma hierarquia interna que separa dispositivos de campo, controladores locais e outros equipamentos que o processo possa requerer, e em cada nível pode haver a utilização de diferentes tipos de redes industriais. Para permitir a integração com o maior número possível de redes e sistemas distribuídos diferentes, optou-se pela utilização do protocolo MODBUS, que é aberto e livre de licenças e, principalmente, a maioria das novas redes possui suporte a dispositivos MODBUS.

O próximo capítulo tratará dos equipamentos e programas utilizados para desenvolver o protótipo do sistema embarcado com controle preditivo.

3 PROTÓTIPO DE SISTEMA EMBARCADO COM MPC

O objetivo deste trabalho é desenvolver um sistema embarcado MPC de baixo custo que possa ser integrado com equipamentos existentes em processos industriais.

O custo final de um sistema embarcado está relacionado ao *hardware* necessário para produzi-lo, às licenças dos programas proprietários utilizados, ao tempo para completar o desenvolvimento e o custo para realização de testes e certificações do comportamento do equipamento. Desta forma, em primeiro lugar, decidiu-se utilizar equipamentos disponíveis a pronta entrega para construir o primeiro protótipo, reduzindo, assim, o tempo de desenvolvimento.

Quanto às licenças de programas proprietários, pode-se evitá-las utilizando *softwares* livres, que são isentos de custos de utilização. Hoje existe uma extensa gama deste tipo de programa que executam as mais diversas tarefas, desde plotagem de gráficos a algoritmos de otimização. Assim, com o uso de *softwares* livres reduz-se ainda mais o custo, pois não é preciso pagar por licenças, e o tempo de desenvolvimento, pois muitos dos algoritmos necessários já estão disponíveis, bastando apenas a sua instalação e configuração.

Neste capítulo, serão discutidas todas as soluções adotadas relativas ao *hardware* e ao *software* do protótipo de sistema embarcado MPC.

3.1 *HARDWARE*

O *hardware* foi escolhido e adquirido pela empresa parceira ATTA [13], e consiste de um kit de desenvolvimento da Technologic Systems [24], intitulado KIT-ARM-7500. Este kit é composto de duas placas de circuito, a TS-7500 e TS-752, protegidas por um envoltório de alumínio modelo TS-ENC750. Na Fig. (8) é mostrado o kit utilizado e na Fig. (9) estão uma vista interna do equipamento, mostrando as placas que o compõem, e uma vista traseira evidenciando seus conectores. As principais características do kit são:

- processador Cavium ARM9 de 250 MHz;
- memória DDR-RAM de 64 Mb;
- entrada micro-SD na qual está conectado um cartão de 2 Gb que

Figura 8 – Kit de desenvolvimento KIT-ARM-7500 utilizado no projeto.

tem a função equivalente a de um disco rígido;

- três Entradas USB: uma para alimentação do equipamento e duas para a conexão de dispositivos externos;
- memória flash on-board de 4 Mb utilizada para o *boot* do sistema operacional;
- possui *hardware* compatível com os seguintes protocolos de comunicação: Ethernet, RS232, RS485 e CANBUS;

Este kit de desenvolvimento possui capacidade de processamento suficiente para executar algoritmos de alto custo computacional, como os de otimização, em tempo hábil, como será mostrado no capítulo 6. Também permite o uso de sistemas operacionais, o que reduz o tempo de desenvolvimento, como será visto a seguir, e ainda é compatível com vários tipos de redes sem a necessidade de se utilizar adaptadores. Devido a estas características e ao preço baixo, este kit foi escolhido para ser o *hardware* do protótipo.

3.2 *SOFTWARE*

Esta seção tratará dos *softwares* utilizados na fase de implementação deste projeto. Serão descritos o Sistema Operacional (SO)

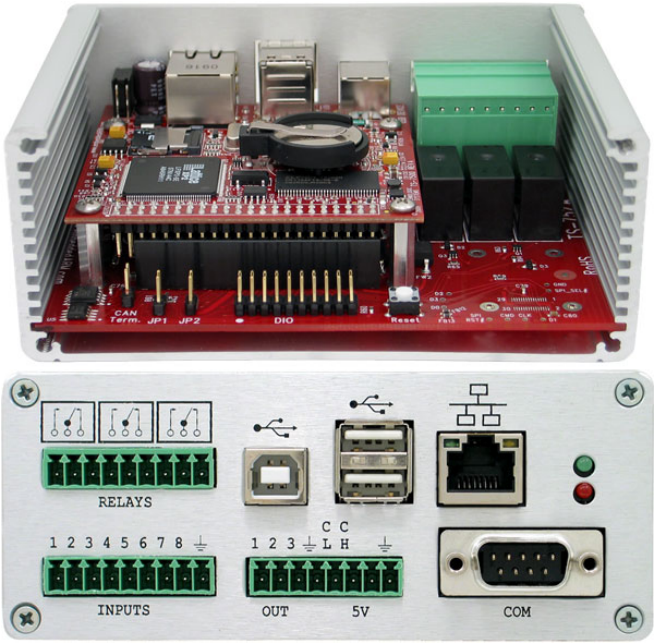


Figura 9 – Vista interna e traseira do kit de desenvolvimento KIT-ARM-7500 utilizado no projeto.

utilizado e a linguagem de programação escolhida para a implementação, assim como as bibliotecas externas instaladas.

3.2.1 Sistema Operacional

Um Sistema Operacional (SO) é um programa que gerencia a utilização do *hardware* de um equipamento e também é responsável por prover uma base para as aplicações do usuário e agir como um intermediário entre os programas e o *hardware* [25]. A Fig. (10) mostra esta organização onde o SO pode ser visto como uma estrutura multi-nível, onde, entre cada nível, existe um "interpretador" que abstrai ou facilita o acesso às funcionalidades da camada imediatamente abaixo [26]. Um SO oferece serviços como: acesso facilitado ao *hardware*, abstração dos detalhes da arquitetura e alocação coordenada dos recursos disponíveis entre as tarefas em execução [4].

O suporte de um SO é fundamental para o processo de desen-

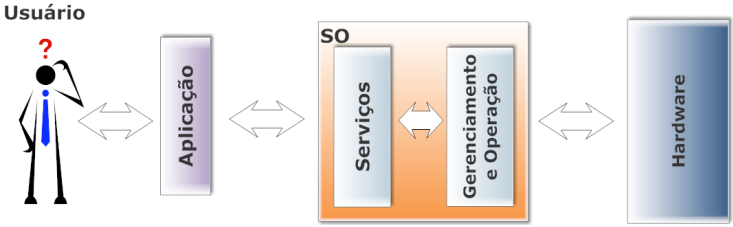


Figura 10 – Esquema da relação entre usuário, aplicação, SO e *hardware* [4].

volvimento de um sistema embarcado. Nesse sentido, o Linux surgiu como uma grande solução para os desenvolvedores, pois é um SO de código aberto, amplamente difundido, que facilita o uso dos recursos de *hardware* do sistema, pois provê abstrações de alto nível como processos concorrentes, *sockets*, sistema de arquivos, grande variedade de *drivers* de dispositivos e documentação extensiva [4].

O kit de desenvolvimento adquirido utiliza o SO Debian Linux (versão do 2.6 do *kernel* Linux), configurado e instalado pelo fornecedor do kit para ser utilizado em sistemas embarcados de baixo consumo de energia. O sistema Debian fornece um ambiente completo de desenvolvimento GNU C/C++, muitos serviços Linux (NFS, FTP, SSH, web server) e várias bibliotecas e programas utilitários [24].

Sistemas embarcados muitas vezes trabalham com restrições de tempo-real, e neste projeto isto é um fator crítico. Os sistemas operacionais que procuram fornecer garantias temporais durante a execução de suas tarefas são chamados Sistemas Operacionais de Tempo Real (SOTR). O *kernel* padrão do Linux, utilizado neste projeto, não pode ser classificado como um sistema determinístico, o que o torna inadequado para aplicações com restrições temporais. Algumas de suas limitações para aplicações de tempo-real são [27]:

- priorização dinâmica: as prioridades dos processos variam ao longo do tempo, o que é ideal para prover acesso justo ao processador a todos os processos. No entanto, pode impedir que um processo urgente acesse o processador assim que necessitar;
- *paging*: mecanismos de paginação podem introduzir atrasos inesperados a menos que a página esteja carregada em memória;
- *coarse-grained synchronization*: Devido ao *kernel* não-interrompível (preemptivo), o sistema pode demorar a responder aos even-

tos, devido a alguma operação que esteja sendo realizada pelo *kernel*.

No entanto, o sistema Linux vem sendo utilizado em várias aplicações cujas restrições temporais são críticas, devido principalmente às extensões criadas ou alterações no próprio *kernel*. A partir do *kernel* 2.6 foram feitas alterações que permitiram caracterizar o Linux como um SO *soft real-time*. Por exemplo, é possível proteger um segmento não interrompível de um programa com *spin locks*, garantindo o acesso da tarefa ao CPU [4]. A denominação *soft real-time* é dada a um SO quando este garante que, na maioria dos casos, os prazos das tarefas serão cumpridos. No caso em que seja necessário garantir deterministicamente o cumprimento das tarefas dentro dos prazos é preciso utilizar um SO *hard real-time*. Existem algumas variantes do Linux e extensões que permitem este tipo de comportamento, tem-se, por exemplo, o Xenomai, RTAI e RTLinux.

No caso deste projeto, a principal tarefa a ser executada é o cálculo da ação de controle pelo algoritmo MPC, que é considerada uma tarefa *soft real-time*, pois a perda dos prazos, definidos pelo tempo de amostragem do processo, não significa uma perda catastrófica para o sistema. Mas é claro que, caso os prazos não sejam obedecidos durante um período suficientemente grande de tempo, poderá haver uma degeneração da resposta do sistema e, no pior caso, instabilidade no processo. Além disso, na versão atual do protótipo, o algoritmo de otimização que realiza o cálculo da ação de controle não é determinístico, ou seja, não é possível saber *a priori* o tempo necessário para a finalização da tarefa. Como também não há como interromper este algoritmo durante sua execução, o que se faz é, através de testes com simuladores, obter o tempo máximo de execução deste cálculo para um determinado ajuste MPC, e daí escolher o tempo de amostragem a ser utilizado, de forma a se ter uma garantia estatística de que, na maior parte do tempo, não haverá perda de prazos. Devido a este fato, o uso de um sistema operacional de tempo-real não traria vantagens, pois, mesmo que todos os recursos fossem utilizados apenas pela tarefa de otimização, não haveria garantias de cumprimento dos prazos.

Em versões futuras deste projeto, pretende-se implementar uma biblioteca própria de otimização, que permitirá um maior controle sobre esta tarefa. Uma discussão melhor do problema de otimização será feito na seção 4.6.2.

No capítulo 6 serão mostrados alguns resultados com relação ao tempo gasto pelo programa do sistema embarcado para calcular e transmitir a ação de controle.

3.2.2 Linguagem de Programação

A linguagem de programação escolhida para implementar o programa desenvolvido para o sistema embarcado foi o Python, uma linguagem interpretada desenvolvida por Guido van Rossum nos anos 80. Por causa de sua natureza interpretada, não é necessário compilar os códigos dos programas, e o usuário pode criá-los e executá-los direto da linha de comando Python [28].

Python foi escolhido por ser uma linguagem de fácil compreensão, altamente portável e disponível em quase todas as plataformas computacionais. Assim, caso se deseje trocar o SO ou mesmo o *hardware* do projeto, não será preciso realizar grandes alterações no programa desenvolvido. Para ilustrar esta característica, pode-se citar este projeto. Uma parte considerável de sua implementação foi feita com Python no SO Windows, mas não houve problemas em se executar os mesmos programas no Linux. O desenvolvimento só passou a ser feito no Linux quando foi preciso implementar e testar funções específicas de comunicação que dependiam do *hardware* do sistema embarcado.

Além disso, há uma comunidade grande de programadores que fornecem bibliotecas Python livres de licenças para serem utilizadas, o que, como já discutido, diminui os custos de desenvolvimento do projeto.

A versão utilizada do Python é a 2.6, e os seguintes pacotes foram instalados:

- NumPy: um pacote de computação científica que fornece implementações de objetos especiais e um número grande de operações matemáticas e lógicas, operações com transformadas de Fourier, álgebra linear, estatística, etc. [29];
- CVXOPT: *Convex Optimization Package*, biblioteca de otimização convexa [30];
- ModbusTK: biblioteca que implementa o protocolo Modbus nas interfaces RS232, RS485 e Ethernet [31].

Uma das vantagens do uso pacote NumPy é a implementação de objetos especiais, por exemplo, matrizes. Este tipo de objeto não possui representação nativa nas linguagens de programação, sendo necessário a criação de classes especiais que abstraem o conceito de matriz e a implementação das operações básicas matriciais. Com o NumPy, a criação de uma matriz e a realização de cálculos ficam resumidos a

alguns comandos, como é mostrado na Fig. (11), onde se cria uma matriz c na linha de comando Python, e depois é feita a operação $d = cc^T$.

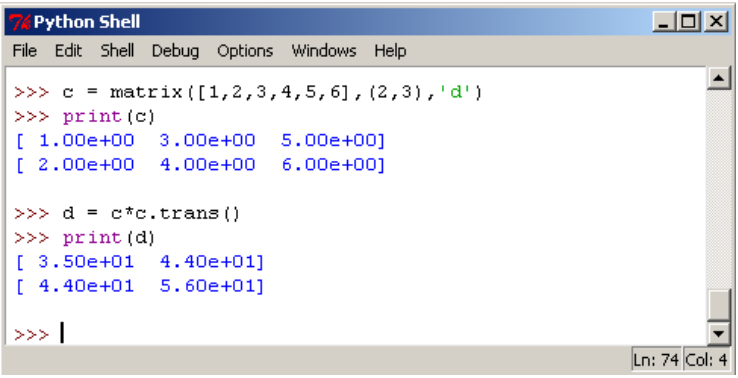


Figura 11 – Criação e realização de uma operação matricial na linha de comando Python.

Outra vantagem da biblioteca NumPy tem relação com a natureza do Python, que é uma linguagem interpretada. Por ser interpretada, a execução de algoritmos matemáticos em Python é geralmente mais lenta do que quando executados em linguagens compiladas como C e C++. A biblioteca NumPy tenta resolver este problema para algoritmos numéricos fornecendo *arrays* multi-dimensionais e funções, e operadores que atuam de forma eficiente em *arrays*. Desta forma, qualquer algoritmo que possa ser expresso primariamente como operações em *arrays* e matrizes pode ser executado quase tão rapidamente quanto seu equivalente em C [32]. Como os algoritmos MPC trabalham essencialmente com operações matriciais, fica claro a vantagem de se utilizar esta biblioteca.

Já o pacote CVXOPT oferece várias funções para resolver diferentes problemas de otimização, mas, para este projeto, o interesse está na resolução de problemas convexos quadráticos da forma

$$\begin{array}{rc} \text{Minimizar:} & J(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{P}\boldsymbol{u} + \boldsymbol{q}^T\boldsymbol{u} + f_0 \\ \text{Sujeito a:} & \overline{\boldsymbol{R}}\boldsymbol{u} \leq \overline{\boldsymbol{c}} \end{array} \tag{3.1}$$

onde deseja-se encontrar o vetor $\boldsymbol{u}$ que minimiza a função custo $J(\boldsymbol{u})$ dadas as matrizes $\boldsymbol{P}$ e $\boldsymbol{q}$, o valor escalar f_0, e as restrições do problema representadas pela desigualdade matricial $\overline{\boldsymbol{R}}\boldsymbol{u} \leq \overline{\boldsymbol{c}}$.

Será visto que o problema de se calcular a ação de controle pe-

los algoritmos MPC pode ser reduzido a um problema quadrático da forma da Eq. (3.1). O algoritmo implementado pelo CVXOPT para resolver este problema é o *Primal-dual Path-Following Method based on Nesterov-Todd scaling* [33]. A Fig. (12) mostra a resolução de um problema quadrático utilizando a função *qp* da biblioteca descrita. Durante a execução do algoritmo de otimização são mostradas as informações pertinentes do problema sendo resolvido e o número de iterações feitas pelo algoritmo.

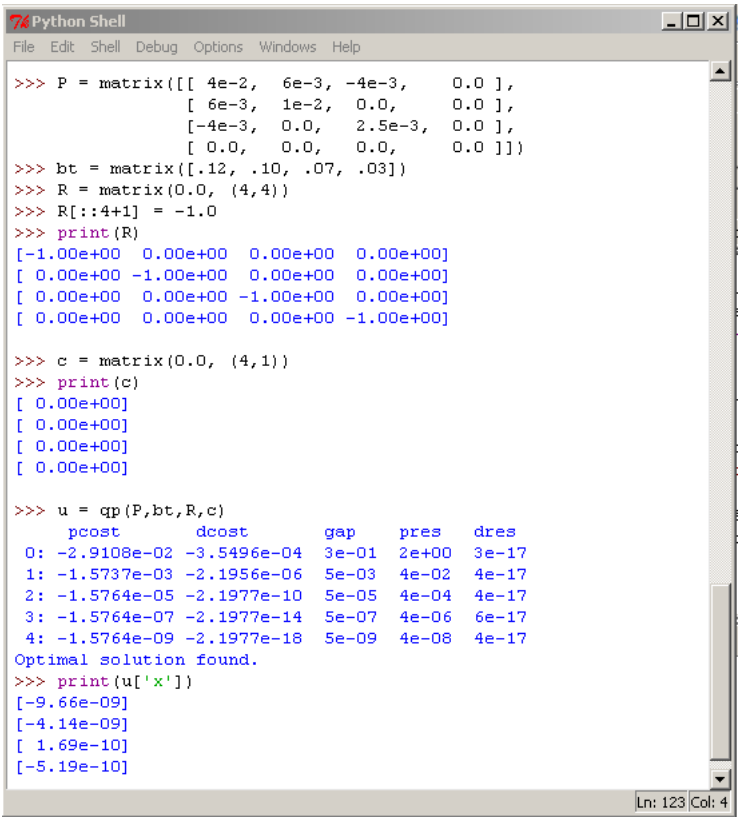


Figura 12 – Resolução de um problema de otimização quadrática utilizando a biblioteca CVXOPT na linha de comando Python.

3.3 ACESSANDO O SISTEMA EMBARCADO

O acesso ao sistema embarcado para execução de aplicativos, transferência de arquivos, instalação de aplicativos, etc., se dá de duas maneiras: (i) através da porta serial; (ii) por SSH através de conexão TCP/IP Ethernet (rede local). O kit de desenvolvimento adquirido possui duas portas seriais RS232, uma delas, chamada RS232/Console, é utilizada somente para ter acesso à linha de comando do SO. Na Fig. (13) é mostrada tela do programa TeraTerm, um programa cliente Telnet gratuito, com a linha de comando do SO após a inicialização do equipamento.

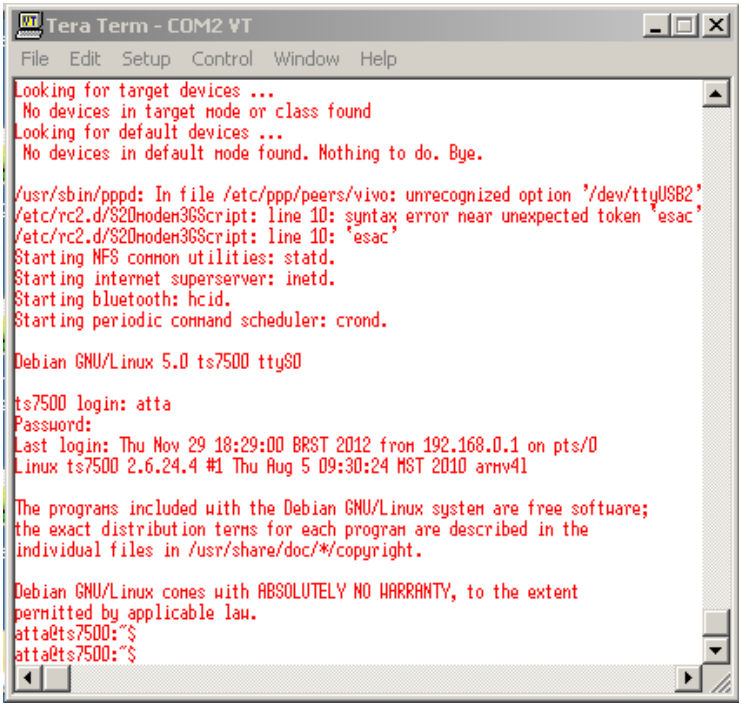


Figura 13 – Acesso à linha de comando do SO do protótipo através do programa TeraTerm.

A segunda alternativa, por rede local, só pode ser utilizada após configurar um endereço IP para o sistema. Depois desta configuração, é possível utilizar qualquer cliente SSH para acessar a linha de comando

e, para transferir arquivos, algum cliente FTP.

3.4 CONCLUSÕES

Neste capítulo foram vistos os equipamentos que compõem o protótipo e também os programas utilizados para realizar a programação e a comunicação com o sistema embarcado. Foi vista também a importância do uso de um sistema operacional assim como as principais bibliotecas utilizadas no desenvolvimento do protótipo.

No capítulo seguinte serão descritos extensivamente os algoritmos MPC utilizados, assim como alguns detalhes de sua implementação.

4 MPC - CONTROLE PREDITIVO BASEADO EM MODELO

A família de algoritmos MPC possui diferentes modos de tratar o processo através de representações diferentes de modelos, mas, basicamente, MPC é um controlador ótimo que minimiza uma função custo que depende da predição futura das saídas do processo, obtidas através de um modelo e dos valores futuros das referências e ações de controle. As diferenças entre os algoritmos estão, principalmente, no tipo de modelo utilizado para representar a planta, nas perturbações consideradas no cálculo das predições futuras e na função custo a ser minimizada [5].

Para calcular a ação de controle os algoritmos executam os seguintes passos:

- predição: baseado no modelo do processo e perturbações, as predições do comportamento futuro da planta são obtidos considerando o estado atual do processo;

- cálculo da ação de controle: o valor da ação de controle a ser aplicada no instante atual é obtido minimizando a função custo;

- atuação: a ação de controle é enviada aos atuadores e, após o período de amostragem, volta-se ao passo de predição.

O modelo de predição varia de acordo com o algoritmo MPC utilizado. O algoritmo MAC (*Model Algorithm Control*) [34] utiliza uma resposta impulsiva, enquanto o DMC (*Dynamic Matrix Control*) [35] e suas variantes utilizam resposta ao degrau. Estes algoritmos são bastante utilizados na prática pois são intuitivos e não precisam de conhecimentos *a priori* do processo para aplicar um método de identificação, e podem ser usados em plantas multivariáveis sem acrescentar complexidade. Por outro lado, apresentam alguns inconvenientes como, por exemplo, não podem ser utilizados em plantas instáveis.

Há também algoritmos que utilizam modelos em espaço de estados como o PFC (*Predictive Functional Control*) [34], e funções de transferência, tais como, o GPC (*Generalized Predictive Control*), o EPSAC (*Extended Prediction Self Adaptive Control*) [36], EHAC (*Extended Horizon Adaptive Control*) [37], entre outros. As representações por espaço de estados e funções de transferência tem como vantagens principais o fato de poderem ser utilizadas em plantas instáveis e que

precisam, em geral, de poucos parâmetros para descrever o comportamento do processo. Outros algoritmos podem ser encontrados em [38].

Como a maioria das aplicações na indústria de processos se baseiam em modelos lineares da planta, assim, na versão atual do protótipo, três algoritmos lineares foram implementados: (a) Controle Preditivo Generalizado (GPC), que utiliza uma representação por funções de transferência do processo; (b) Controle Preditivo Generalizado com Compensação de Atraso de Transporte (DTCGPC), uma versão modificada do GPC para processos com atraso que permite um ajuste mais fácil da robustez do sistema em malha fechada; (c) Controle Preditivo Baseado em Modelo por Espaço de Estados (SSMPC), que utiliza modelos por espaço de estados para representar o processo. Em versões futuras do sistema, planeja-se implementar estratégias MPC não-lineares tais como a proposta em [39].

Em todos os casos o algoritmo é implementado usando o conceito de resposta livre e forçada, ambas obtidas através dos modelos do processo e perturbações. A resposta livre do sistema representa a dinâmica de malha aberta deste caso não haja alterações nas ações de controle, e a forçada está relacionada com a dinâmica caso haja variações de controle. Assim, para um sistema multivariável com m entradas e n saídas, o vetor de predições futuras das saídas do processo $\hat{\boldsymbol{y}}$ pode ser representado por:

$$\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{u}(t) + \boldsymbol{f}, \tag{4.1}$$

onde $\boldsymbol{f}$ é o vetor de resposta livre, $\boldsymbol{u}(t)$ é o vetor de incrementos futuros das ações de controle que serão calculados e $\boldsymbol{H}$ é uma matriz obtida através do modelo da planta. Nesta equação $\hat{\boldsymbol{y}}$ e $\boldsymbol{f}$ tem dimensão $n_y \times 1$, $\boldsymbol{u}(t)$ tem dimensão $n_u \times 1$ e $\boldsymbol{H}$ é $n_y \times n_u$, onde $n_y = \sum_{i=1}^{m} N_i$, $n_u = \sum_{i=1}^{n} N_{u_i}$, N_i é o horizonte de predição da saída y_i e N_{u_i} é o horizonte de controle da entrada u_i.

A função custo mais comum encontrada nos algoritmos MPC é quadrática e assume a forma

$$J = [\boldsymbol{w} - \hat{\boldsymbol{y}}]^T \boldsymbol{Q_y} [\boldsymbol{w} - \hat{\boldsymbol{y}}] + \boldsymbol{u}^T \boldsymbol{Q_u} \boldsymbol{u}, \tag{4.2}$$

onde $\boldsymbol{w}$ é o vetor de referências futuras e $\boldsymbol{Q_y}$, $\boldsymbol{Q_u}$ são, respectivamente, as matrizes de ponderação dos erros e das ações de controle. Neste problema, os parâmetros de projeto do controlador são os horizontes e as matrizes de ponderação.

Substituindo a Eq. (4.1) na Eq. (4.2) e rearranjado os termos,

obtém-se

$$J(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{P}\boldsymbol{u} + \boldsymbol{q}^T\boldsymbol{u} + f_0, \tag{4.3}$$

onde

$$\begin{aligned} \boldsymbol{P} &= 2(\boldsymbol{H}^T\boldsymbol{Q_y}\boldsymbol{H} + \boldsymbol{Q_u}), \\ \boldsymbol{q}^T &= 2(\boldsymbol{f} - \boldsymbol{w})^T\boldsymbol{Q_y}\boldsymbol{H}, \\ f_0 &= (\boldsymbol{f} - \boldsymbol{w})^T\boldsymbol{Q_y}(\boldsymbol{f} - \boldsymbol{w}). \end{aligned}$$

Levando em conta as restrições do processo, o problema a ser resolvido pelo algoritmo MPC pode ser representado como o seguinte problema de otimização quadrática:

$$\begin{array}{rc} \text{Minimizar:} & J(\boldsymbol{u}) \\ \text{Sujeito a:} & \overline{\boldsymbol{R}}\boldsymbol{u} \leq \overline{\boldsymbol{c}} \end{array} \tag{4.4}$$

onde $\overline{\boldsymbol{R}}\boldsymbol{u} \leq \overline{\boldsymbol{c}}$ representa as restrições nas entradas e saídas do processo.

Através da escolha dos horizontes de predição e controle, e das ponderações, define-se qual será o ajuste do controlador MPC. Apesar de os conceitos por trás destes parâmetros serem intuitivos, não é trivial encontrar uma relação entre seus valores e o controlador resultante. Desta forma, geralmente, o que se faz é, através de simulações, ajustar os parâmetros para se obter uma dinâmica especificada e daí aplicar o controle de fato. Os parâmetros de ajuste são escolhidos considerando os seguintes conceitos [5]:

- Horizontes de predição: estes parâmetros definem quais são os instantes de tempo em que é importante que a saída siga a referência. Assim, se for escolhido um horizonte de predição inicial grande, significa que os erros nos primeiros instantes não são importantes e serão desconsiderados. Isto originará uma resposta mais suave do processo. Note que se o processo possuir um atraso de transporte d, não faz sentido escolher o horizonte inicial menor que $t + d$, porque a saída só será afetada pelas alterações de controle em t após $t + d$. Além disso, se o processo é de fase não-mínima, este parâmetro permite que os primeiros instantes de tempo, onde a resposta é inversa, sejam ignorados.

- Ponderações: as ponderações dos erros e dos incrementos de controle podem ser escolhidos como constantes em todo o horizonte. Assim, no caso monovariável, se as ponderações do controle forem maior do que a dos erros, significa que a ação de controle terá uma

variação mais suave, tornando a resposta do sistema mais lenta. Pode-se adotar também, por exemplo, uma ponderação exponencial onde os erros iniciais tem mais peso do que os erros futuros, desta forma o controlador tentará agir de forma a minimizar o máximo possível os erros iniciais.

Nas seções seguintes deste capítulo serão descritos os algoritmos MPC implementados e também serão discutidas algumas ferramentas matemáticas utilizadas por eles: as equações diofantinas e a representação de sistemas multivariáveis por matrizes fracionárias. Além da descrição geral de cada algoritmo, serão dados os pseudo-códigos contendo os passos para a execução de cada um. Há também uma seção sobre como se obtém uma representação matricial das restrições comumente encontradas nos processos industriais.

4.1 EQUAÇÕES DIOFANTINAS

Equações diofantinas polinomiais são equações onde as variáveis incógnitas a serem determinadas são polinômios. Este tipo de equação pode ser representada da seguinte forma:

$$S(z^{-1})A(z^{-1}) + T(z^{-1})B(z^{-1}) = C(z^{-1}) \tag{4.5}$$

onde todas as variáveis são polinômios e $A(z^{-1})$, $B(z^{-1})$ e $C(z^{-1})$ são conhecidos, faltando apenas determinar $S(z^{-1})$ e $T(z^{-1})$. Esta equação será utilizada extensivamente durante o desenvolvimento do algoritmo GPC e DTCGPC.

No caso particular dos algoritmos preditivos, os polinômios utilizados são função do operador de atraso discreto z^{-1}, por exemplo, $A(z^{-1}) = a_0 + a_1 z^{-1} + \ldots + a_n z^{-n}$. O objetivo de se utilizar equações diofantinas nos algoritmos preditivos é separar o polinômio original $A(z^{-1})$ em dois novos polinômios $S(z^{-1})$ e $T(z^{-1})$ de tal forma que a predição sendo calculada possa ser separada em termos dependentes, respectivamente, das entradas futuras e passadas. Neste caso particular, o polinômio $B(z^{-1})$ assume sempre a forma $B(z^{-1}) = z^{-k}$, onde $k > 0$ e depende da predição sendo calculada k amostras à frente do tempo atual.

Uma interpretação que se pode fazer, neste caso, é que na solução da equação diofantina, os polinômios $S(z^{-1})$ e $T(z^{-1})$ são, respectivamente, o quociente e o resto da divisão do polinômio $C(z^{-1})$ por $A(z^{-1})$. Por exemplo, dado que $C(z^{-1}) = 1$, $A(z^{-1}) = 1 - 0,1z^{-1}$ e

$$
\begin{array}{rl}
1 & \left|\, 1-0.1z^{-1} \right. \\
\cline{2-2}
-1+0.1z^{-1} & 1+0.1z^{-1} \\
\cline{1-1}
\quad 0.1z^{-1} & \\
\quad -0.1z^{-1}+0.01z^{-2} & \\
\cline{1-1}
\qquad\qquad 0.01z^{-2} &
\end{array}
$$

Figura 14 – Obtenção da solução de uma equação diofantina.

$k = 2$, o que implica $B(z^{-1}) = z^{-2}$, a solução é obtida realizando k passos de divisão polinomial, como mostrado na Fig. (14), resultando nos polinômios $T(z^{-1}) = 0,01$ e $S(z^{-1}) = 1 + 0,1z^{-1}$. Note que, como os polinômios estão em função de z^{-1}, poderia-se realizar infinitos passos de divisão.

Para a solução deste tipo de equação em um programa de computador, é importante saber *a priori* as ordens dos polinômios resultantes, e isto pode ser obtido analisando-se os polinômios conhecidos. Define-se a ordens dos polinômios $A(z^{-1})$, $B(z^{-1})$, $C(z^{-1})$, $S(z^{-1})$ e $T(z^{-1})$ como, respectivamente na, nb, nc, ns e nt. A ordem de $S(z^{-1})$ depende unicamente da quantidade de passos realizados na divisão polinomial, ou seja, de k, assim, $ns = k - 1$. Já nt depende dos polinômios $A(z^{-1})$ e $C(z^{-1})$. Para satisfazer a equação diofantina, se $nc > ns + na$, então $nt + nb = nc$, caso contrário, $nt + nb = ns + na$.

Definindo o tamanho de um polinômio como a quantidade de coeficientes que este possui, ou seja, $len(C(z^{-1})) = nc + 1$, o algoritmo para a solução da equação diofantina citada anteriormente é:

- Passo 1: Calcula-se as ordens de $S(z^{-1})$ e $T(z^{-1})$.
  - $nb = k$
  - $ns = k - 1$
  - Se $nc > ns + na$
    Então
    $nt = nc - nb = nc - k$
    Senão
    $nt = ns + na - nb = (k - 1) + na - k = na - 1$
- Passo 2: Iniciar os vetores vs e vt com tamanho $len(S(z^{-1}))$ e $len(T(z^{-1}))$, respectivamente.
- Passo 3: Executar um passo da divisão de $C(z^{-1})$ por $A(z^{-1})$ obtendo o primeiro coeficiente de $S(z^{-1})$, que é armazenado em vs.

Armazenar os coeficientes do polinômio resto $Q(z^{-1})$ da divisão em vq.

- Passo 4: Executar $k-1$ vezes:
  - Dividir Q por A, obtendo um novo coeficiente de S
  - Armazenar em vq os coeficientes do novo polinômio resto Q
- Passo 5: Terminado o passo anterior, $T(z^{-1}) = Q(z^{-1})$, assim, os coeficientes de $T(z^{-1})$ já estão armazenados em vq

4.2 DMF - DESCRIÇÃO MATRICIAL FRACIONÁRIA

A matriz de transferência é a representação clássica de processos multivariáveis. Isto porque estas matrizes podem ser obtidas por análises frequenciais ou obtendo as respostas ao impulso ou ao degrau da planta (nos casos estáveis). Na maioria dos processos industriais, onde plantas estáveis de baixa ordem são comuns, qualquer coluna da matriz de transferência pode ser obtida aplicando um degrau na entrada correspondente e medindo-se o ganho estático, a constante de tempo e o atraso para cada uma das saídas. Também podem ser usados, por exemplo, algoritmos de identificação por mínimos quadrados para a obtenção das funções de transferência. Repetindo-se o processo para cada uma das entradas, obtém-se a matriz completa [5]. A Eq. (4.6) dá uma matriz de transferência para um sistema genérico com m entradas e n saídas, ou seja,

$$\mathbf{\Gamma}(z^{-1}) = \begin{bmatrix} \frac{N_{11}(z^{-1})}{D_{11}(z^{-1})} & \cdots & \frac{N_{1m}(z^{-1})}{D_{1m}(z^{-1})} \\ \vdots & \ddots & \vdots \\ \frac{N_{n1}(z^{-1})}{D_{11}(z^{-1})} & \cdots & \frac{N_{nm}(z^{-1})}{D_{nm}(z^{-1})} \end{bmatrix} \tag{4.6}$$

onde N_{ij} e D_{ij} representam, respectivamente, o numerador e o denominador da função de transferência da entrada j para a saída i.

É possível decompor $\mathbf{\Gamma}(z^{-1})$ da seguinte forma

$$\mathbf{\Gamma}(z^{-1}) = \boldsymbol{A}(z^{-1})^{-1}\boldsymbol{B}(z^{-1}). \tag{4.7}$$

onde $\boldsymbol{A}(z^{-1})$ e $\boldsymbol{B}(z^{-1})$ são matrizes polinomiais de ordem, respectivamente $n \times n$ e $n \times m$.

Esta representação é chamada Descrição Matricial Fracionária (DMF) ou, em inglês, *Matrix Fraction Description* (MFD). Esta de-

composição será utilizada para a obtenção dos modelos utilizados nos algoritmos MPC multivariáveis descritos nas próximas seções.

O modo mais simples de se obter a DMF é fazer com que $\boldsymbol{A}(z^{-1})$ seja diagonal com seus elementos iguais aos mínimos múltiplos comuns dos denominadores das linhas correspondentes de $\boldsymbol{\Gamma}(z^{-1})$. A matriz $\boldsymbol{B}(z^{-1})$ é então igual a $\boldsymbol{B}(z^{-1}) = \boldsymbol{A}(z^{-1})\boldsymbol{\Gamma}(z^{-1})$.

O algoritmo para obter a DMF, dada uma matriz de transferência $\boldsymbol{\Gamma}(z^{-1})$, é descrito a seguir:

- Passo 1: Para cada linha i de $\boldsymbol{\Gamma}(z^{-1})$, calcular o polinômio $A_i(z^{-1})$ que é igual ao mínimo múltiplo comum entre os denominadores da linha correspondente, obtendo-se a matriz $\boldsymbol{A}(z^{-1})$

- Passo 2: Calcular a matriz $\boldsymbol{B}(z^{-1})$ dado que:

  – $B_{ij}(z^{-1}) = \frac{A_i(z^{-1})N_{ij}(z^{-1})}{D_{ij}(z^{-1})}$

Como por exemplo, considere a seguinte matriz de transferência:

$$\boldsymbol{\Gamma}(z^{-1}) = \begin{bmatrix} \frac{0{,}0420z^{-1}}{1-0{,}9580z^{-1}} & \frac{0{,}4758z^{-1}}{1-0{,}9048z^{-1}} \\ \\ \frac{0{,}0582z^{-1}}{1-0{,}9418z^{-1}} & \frac{0{,}1445z^{-1}}{1-0{,}9277z^{-1}} \end{bmatrix}, \tag{4.8}$$

e aplicando-se o algoritmo descrito anteriormente, obtêm-se as seguintes matrizes que representam a DMF

$$\begin{aligned} \boldsymbol{A}(z^{-1}) &= \begin{bmatrix} D_{11}(z^{-1})D_{12}(z^{-1}) & 0 \\ \\ 0 & D_{21}(z^{-1})D_{22}(z^{-1}) \end{bmatrix} \\ &= \begin{bmatrix} 1-1{,}8629z^{-1}+0{,}8669z^{-2} & 0 \\ \\ 0 & 1-1{,}8695z^{-1}+0{,}8737z^{-2} \end{bmatrix}, \end{aligned}$$

$$\begin{aligned} \boldsymbol{B}(z^{-1}) &= \begin{bmatrix} \frac{A_1(z^{-1})(0{,}0420z^{-1})}{1-0{,}9580z^{-1}} & \frac{A_1(z^{-1})(0{,}4758z^{-1})}{1-0{,}9048z^{-1}} \\ \\ \frac{A_2(z^{-1})(0{,}0582z^{-1})}{1-0{,}9418z^{-1}} & \frac{A_2(z^{-1})(0{,}1445z^{-1})}{1-0{,}9277z^{-1}} \end{bmatrix} \\ &= \begin{bmatrix} 0{,}0420z^{-1}-0{,}0380z^{-2} & 0{,}4758z^{-1}-0{,}4559z^{-2} \\ \\ 0{,}0582z^{-1}-0{,}0540z^{-2} & 0{,}1445z^{-1}-0{,}1361z^{-2} \end{bmatrix}. \end{aligned}$$

4.3 GPC - CONTROLE PREDITIVO GENERALIZADO

O algoritmo preditivo chamado de GPC, do inglês *Generalized Predictive Control*, proposto em [40], é um dos mais populares algoritmos preditivos tanto na indústria como no meio acadêmico [5]. Ao contrário de outros algoritmos, o GPC consegue lidar com plantas instáveis e de fase não-mínima, além de considerar a ponderação dos incrementos de controle em sua função custo. O GPC também permite uma liberdade maior na sintonia do controlador devido ao número de parâmetros de ajuste. Devido a esta liberdade, alguns dos outros algoritmos MPC são considerados casos particulares do GPC [5].

Inicialmente será apresentada a formulação do GPC para um sistema SISO (uma entrada e uma saída), depois esta será estendida para o caso multivariável.

4.3.1 Formulação para o Caso SISO

Considerando-se nesta seção que a planta opera próximo de um ponto de operação e que pode ser representada por um modelo linear discreto do tipo:

$$A(z^{-1})y(t) = B(z^{-1})u(t-1-d)+D(z^{-1})v(t-dv)+C(z^{-1})\frac{e(t)}{\Delta} \quad (4.9)$$

onde $u(t)$ representa a entrada, $v(t)$ uma perturbação mensurável, $y(t)$ a saída, d e dv representam os atrasos de transporte da entrada e perturbação, respectivamente, $e(t)$ é um ruído branco de média zero e $\Delta = 1 - z^{-1}$ é o operador diferença. A, B, D e C representam polinômios no operador de atraso z^{-1}:

$$\begin{aligned}
A(z^{-1}) &= 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_{na} z^{-na} && (4.10)\\
B(z^{-1}) &= b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_{nb} z^{-nb} && (4.11)\\
D(z^{-1}) &= d_0 + d_1 z^{-1} + d_2 z^{-2} + \ldots + d_{nd} z^{-nd} && (4.12)\\
C(z^{-1}) &= 1 + c_1 z^{-1} + c_2 z^{-2} + \ldots + c_{nc} z^{-nc} && (4.13)
\end{aligned}$$

Este tipo de modelo linear é denominado CARIMA, do inglês *Controlled Auto-Regressive and Integrated Moving Average*, onde as perturbações não-mensuráveis, isto é, a diferença entre a saída do mo-

delo e a do sistema são representadas pelo termo

$$\frac{C(z^{-1})e(t)}{\Delta}$$

da Eq. (4.9). Foi mostrado que este termo permite representar bem mudanças aleatórias, *off-sets* e outros fenômenos normalmente encontrados nos meios industriais [40]. A inclusão deste modelo de perturbação permite garantir erro nulo para perturbações de carga do tipo degrau. O polinômio $C(z^{-1})$ é utilizado para modelar as características estocásticas das perturbações, mas este polinômio é de difícil estimação devido à sua característica variante no tempo e à dificuldade do modelo CARIMA de descrever perturbações determinísticas. Assim, é comum considerá-lo igual a 1 ou, como será visto posteriormente, substituí-lo pelo polinômio $T(z^{-1})$, um parâmetro projetado para aumentar a robustez do sistema em malha fechada.

O algoritmo GPC consiste em aplicar uma sequência de controle que minimize uma função custo da forma:

$$\begin{aligned} J(\boldsymbol{u}) &= \sum_{j=N_1}^{N_2} \delta(j)[\hat{y}(t+j|t) - w(t+j)]^2 + \\ &\quad \sum_{j=1}^{N_u} \lambda(j)[\Delta u(t+j-1)]^2 \end{aligned} \tag{4.14}$$

onde $\hat{y}(t+j|t)$ é a predição ótima da saída do sistema j instantes de tempo à frente do tempo atual t, considerando os dados existentes no tempo t. N_1 e N_2 são os horizontes mínimos e máximos considerados na função custo, N_u é o horizonte de controle, $w(t+j)$ é a trajetória de referência futura e, $\lambda(j)$ e $\delta(j)$ são, respectivamente, os pesos da ação de controle e dos erros futuros nos instantes de tempo correspondentes. É possível estipular diferentes pesos para cada erro futuro e ação de controle j, mas é mais comum estipular pesos fixos em todo o horizonte. O horizonte de predição N_1 é geralmente igual a 1 mas, quando existe um atraso d entre a entrada e a saída do processo, o horizonte assume o valor $N_1 = d + 1$. Isto é intuitivamente óbvio, já que as alterações do sinal de controle feitas após o instante de tempo t só afetaram as predições do sistema após o instante $t + d + 1$.

Ao minimizar a função custo, obtém-se uma sequência de sinais de controle $\boldsymbol{\Delta u} = [\Delta u(t), \ldots, \Delta u(t+N_u-1)]^T$, que fará com que a saída do sistema $y(t)$ se aproxime da referência $w(t)$ de forma ótima. Para

realizar esta minimização, deve-se primeiro obter as predições futuras que são dadas por

$$y(t+j) = \frac{B}{A}(z^{-1})u(t-1-d+j)+\frac{D}{A}(z^{-1})v(t-dv+j)+\frac{T}{\Delta A}(z^{-1})e(t+j). \tag{4.15}$$

Considere agora a seguinte equação diofantina:

$$T(z^{-1}) = E_j(z^{-1})\Delta A(z^{-1}) + z^{-j}F_j(z^{-1}) \tag{4.16}$$

onde E_j e F_j são polinômios em z^{-1} e j está relacionado com a predição sendo calculada. A ordem de E_j é $nE = j-1$, assim, considerando a igualdade polinomial, a ordem de F_j, nF, depende das ordens dos polinômios T e A. Se $nT > na + j$, então $nF = nT - j$, caso contrário, $nF = nA$.

Substituindo a Eq. (4.16) na Eq. (4.15) tem-se

$$\begin{aligned} y(t+j) &= \frac{B}{A}u(t-1-d+j) + \frac{D}{A}(z^{-1})v(t-dv+j) \\ &\quad + \frac{E_j\Delta A + z^{-j}F_j}{\Delta A}e(t+j) \\ &= \frac{B}{A}u(t-1-d+j) + \frac{D}{A}(z^{-1})v(t-dv+j) \\ &\quad + E_j e(t+j) + \frac{F_j}{\Delta A}e(t). \end{aligned} \tag{4.17}$$

Do modelo da Eq. (4.9) tem-se que

$$e(t) = \frac{\Delta A}{T}y(t) - \frac{B}{T}\Delta u(t-1-d) - \frac{D}{T}\Delta v(t-dv). \tag{4.18}$$

Substituindo a Eq. (4.18) na Eq. (4.17) e rearranjando os termos, obtém-se

$$\begin{aligned} y(t+j) &= \frac{F_j}{T}y(t) + \left(1 - \frac{z^{-j}F_j}{T}\right)\frac{B}{A}u(t-1-d+j) \\ &\quad + \left(1 - \frac{z^{-j}F_j}{T}\right)\frac{D}{A}v(t-dv+j) \\ &\quad + E_j e(t+j). \end{aligned} \tag{4.19}$$

Da equação diofantina dada na Eq. (4.16), tem-se que

$$\frac{E_j \Delta A}{T} = 1 - \frac{z^{-j} F_j}{T}, \tag{4.20}$$

Substituindo a Eq. (4.20) na Eq. (4.19) obtém-se a equação que calcula a predição da saída do sistema no tempo $t + j$ em função das saídas passadas e dos incrementos, passados e futuros, das entradas e perturbações:

$$\begin{aligned} y(t+j|t) &= \frac{F_j}{T} y(t) + \frac{E_j B}{T} \Delta u(t-1-d+j) \\ &\quad + \frac{E_j D}{T} \Delta v(t-dv+j) \\ &\quad + E_j e(t+j). \end{aligned} \tag{4.21}$$

Como foi mostrado, a resposta do sistema pode ser dividida em livre e forçada, sendo que aquela representa o comportamento do sistema caso não houvesse alteração nos sinais de controle e a forçada representa a resposta com alterações. É interessante fazer esta separação na Eq. (4.21), pois assim se obtêm de forma clara como a variação futura do controle afetará a predição. Para fazer esta separação, define-se duas novas equações diofantinas, a Eq. (4.22) é aplicada no caso das entradas e a Eq. (4.23) no caso as perturbações:

$$E_j B(z^{-1}) = H_j T + z^{-k} I_j \tag{4.22}$$

$$E_j D = H_{v_j} T + z^{-k} I_{v_j} \tag{4.23}$$

onde $k = j - d$, na primeira equação e $k = j - dv$ na segunda, j faz referência à predição sendo calculada, H_j e H_{v_j} são os polinômios dependentes das ações de controle e perturbações futuras, I_j e I_{v_j} são os polinômios relacionados aos incrementos de controle e perturbações passados. O polinômio no lado esquerdo da Eq. (4.22) tem ordem $neb = j - 1 + nb$, e, como $nH = k - 1 = j - d - 1$, se $neb > nH + nT$, a ordem de I_j é obrigatoriamente $nI = nb + d - 1$, caso contrário, tem-se que $nI = nT - 1$, o mesmo raciocínio vale para as ordens dos polinômios na Eq. (4.23).

A variável k serve para corrigir o cálculo dos polinômios H e I quando há a presença de atraso. Por exemplo, dado um sistema MISO com atrasos nas entradas e perturbações, e dado que o atraso mínimo entre as entradas é d_{min}, o horizonte inicial deve ser escolhido

$N_1 = d_{min} + 1$ pois não faz sentido, como explicado anteriormente, escolher N_1 menor que o atraso mínimo. Se uma entrada u_i tiver um valor de atraso $d_i > d_{min}$, isso significará que a predição $\hat{y}(t+N_1|t)$ só dependerá dos valores passados desta variável, assim não fará sentido utilizar a Eq. (4.22) para separar os incrementos Δu_i entre passados e futuros. Este fato é refletido pela condição $k <= 0$.

Substituindo as equações diofantinas Eq. (4.22) e Eq. (4.23) na Eq. (4.21), e dado que o termo $E_j e(t+j)$ só depende de valores futuros do ruído, pois $ne = j - 1$, e que o valor esperado do ruído no futuro é zero, obtém-se a predição ótima $\hat{y}(t+j)$ na Eq. (4.24).

$$\begin{aligned}\hat{y}(t+j|t) &= H_j \Delta u(t-1-d+j) + I_j \frac{\Delta u(t-1)}{T} \\ &\quad + H_{v_j} \Delta v(t-d_v+j) + I_{v_j} \frac{\Delta v(t)}{T} \\ &\quad + \frac{F_j}{T} y(t) \end{aligned} \tag{4.24}$$

As predições ótimas da saída entre os horizontes N_1 e N_2 podem ser rearranjadas em forma matricial, isto é,

$$\begin{aligned}\hat{\boldsymbol{y}} &= \boldsymbol{H}\boldsymbol{u}(t) + \boldsymbol{H_v}(z^{-1})\Delta v(t+1) + \boldsymbol{I}(z^{-1})\Delta u_f(t-1) \\ &\quad + \boldsymbol{I_v}(z^{-1})\Delta v_f(t) + \boldsymbol{F}(z^{-1}) y_f(t)\end{aligned} \tag{4.25}$$

onde

$$\hat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}(t+N_1|t) \\ \vdots \\ \hat{y}(t+N_2|t) \end{bmatrix} \qquad \boldsymbol{u}(t) = \begin{bmatrix} \Delta u(t) \\ \vdots \\ \Delta u(t+N_u) \end{bmatrix}$$

$$\boldsymbol{I}(z^{-1}) = \begin{bmatrix} I_{N_1}(z^{-1}) \\ \vdots \\ I_{N_2}(z^{-1}) \end{bmatrix} \qquad \boldsymbol{I_v}(z^{-1}) = \begin{bmatrix} I_{v_{N_1}}(z^{-1}) \\ \vdots \\ I_{v_{N_2}}(z^{-1}) \end{bmatrix}$$

$$\boldsymbol{F}(z^{-1}) = \begin{bmatrix} F_{N_1}(z^{-1}) \\ \vdots \\ F_{N_2}(z^{-1}) \end{bmatrix} \qquad \boldsymbol{H_v}(z^{-1}) = \begin{bmatrix} H_{v_{N_1}}(z^{-1}) \\ \vdots \\ H_{v_{N_2}}(z^{-1}) \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} h_0 & 0 & \dots & 0 \\ h_1 & h_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ h_{N-2} & h_{N-3} & \dots & h_{N-1-N_u} \\ h_{N-1} & h_{N-2} & \dots & h_{N-N_u} \end{bmatrix},$$

Δu_f, v_f, y_f são, respectivamente, as variáveis de entrada, perturbação e saída filtradas pelo polinômio T, e $N = N_2 - N_1 + 1$, ou seja, o número de predições de saída. Na Eq. (4.25), nota-se a dependência nos valores futuros da perturbação mensurável $v(t)$ no termo $\boldsymbol{H_v}(z^{-1})v(t+1)$. Este termo só é utilizável quando é possível saber, ou estimar, com antecedência o valor da perturbação no futuro como, por exemplo, no caso em que $v(t)$ é a referência de um processo secundário.

Agrupando os termos da Eq. (4.25) dependentes apenas do passado e o termo $\boldsymbol{H_v(z^{-1})}v(t+1)$, que depende das variações futuras da perturbação, em $\boldsymbol{f}$, o vetor da resposta livre, tem-se a seguinte expressão compacta das predições:

$$\hat{\boldsymbol{y}} \quad = \quad \boldsymbol{H}\boldsymbol{u}(t) + \boldsymbol{f}. \tag{4.26}$$

A função custo na Eq. (4.14) pode ser reescrita na seguinte forma matricial:

$$J = (\hat{\boldsymbol{y}} - \boldsymbol{w})^T \boldsymbol{\delta} (\hat{\boldsymbol{y}} - \boldsymbol{w}) + \boldsymbol{\lambda}\boldsymbol{u}^T\boldsymbol{u}, \tag{4.27}$$

onde $\boldsymbol{w} = [w(t+N_1), \dots, w(t+N_2)]^T$ e, $\boldsymbol{\lambda}$ e $\boldsymbol{\delta}$, são matrizes diagonais de ordem, respectivamente, $N_u \times N_u$ e $(N_2 - N_1 + 1) \times (N_2 - N_1 + 1)$, onde o elemento (i,i) é o peso do termo $\Delta u(t+i-1)$ no caso de $\boldsymbol{\lambda}$ e o peso de $\hat{y}(t+N_1+i-1)$ no caso de $\boldsymbol{\delta}$. Substituindo a Eq. (4.26) na Eq. (4.27) tem-se:

$$J = (\boldsymbol{H}\boldsymbol{u} + \boldsymbol{f} - \boldsymbol{w})^T \boldsymbol{\delta} (\boldsymbol{H}\boldsymbol{u} + \boldsymbol{f} - \boldsymbol{w}) + \boldsymbol{\lambda}\boldsymbol{u}^T\boldsymbol{u} \tag{4.28}$$

Rearranjando a Eq. (4.28) para a forma padrão quadrática:

$$J = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{P}\boldsymbol{u} + \boldsymbol{q}^T\boldsymbol{u} + f_0 \tag{4.29}$$

onde

$$\begin{aligned} \boldsymbol{P} &= 2(\boldsymbol{H}^T\boldsymbol{\delta}\boldsymbol{H} + \boldsymbol{\lambda}), \\ \boldsymbol{q}^T &= 2(\boldsymbol{f} - \boldsymbol{w})^T\boldsymbol{\delta}\boldsymbol{H}, \\ f_0 &= (\boldsymbol{f} - \boldsymbol{w})^T\boldsymbol{\delta}(\boldsymbol{f} - \boldsymbol{w}). \end{aligned}$$

No caso sem restrições, a minimização da função custo na Eq. (4.28) pode ser feita de forma analítica através de derivação matricial [5], resultando no seguinte vetor de controles ótimos:

$$\begin{aligned} \boldsymbol{u} &= (\boldsymbol{H}^T\boldsymbol{\delta}\boldsymbol{H} + \boldsymbol{\lambda})^{-1}\boldsymbol{H}^T\boldsymbol{\delta}(\boldsymbol{w} - \boldsymbol{f}) \\ &= \boldsymbol{K}(\boldsymbol{w} - \boldsymbol{f}) \end{aligned} \tag{4.30}$$

onde $\boldsymbol{K} = (\boldsymbol{H}^T\boldsymbol{\delta}\boldsymbol{H} + \boldsymbol{\lambda})^{-1}\boldsymbol{H}^T\boldsymbol{\delta}$. Isto tem um significado claro, se não há erros preditos futuros, ou seja, $\boldsymbol{w} - \boldsymbol{f} = 0$, então não é preciso alterar o controle, já que o objetivo será atingido com a resposta livre do sistema. Caso haja erros futuros, haverá um incremento na ação de controle proporcional (por um fator de $\boldsymbol{K}$) a este erro [5]. Note que, seguindo a ideia do horizonte deslizante, somente o primeiro elemento de $\boldsymbol{u}$, que equivale ao incremento de controle $\Delta u(t)$, será utilizado, pois no instante $t + 1$ os cálculos serão refeitos levando em conta os novos dados provenientes do processo. Assim, é possível simplificar o cálculo da ação de controle da seguinte maneira:

$$\Delta u(t) = K_1(\boldsymbol{w} - \boldsymbol{f}) \tag{4.31}$$

$$u(t) = u(t-1) + \Delta u(t) \tag{4.32}$$

onde K_1 é a primeira linha da matriz $\boldsymbol{K}$, e tem ordem $1 \times N$.

4.3.2 Formulação para o Caso MIMO

O algoritmo MIMO-GPC, assim como no caso SISO, calcula a ação de controle a partir da otimização de uma função custo. Para um sistema MIMO com m entradas e n saídas, esta função é dada por:

$$J = \sum_{i=1}^{n} \sum_{j=N_{1i}}^{N_{2i}} \delta_i(j)[\hat{y}_i(t+j|t) - w_i(t+j)]^2 + \sum_{i=1}^{m} \sum_{j=1}^{N_{ui}} \lambda_i(j)[\Delta u_i(t+j-1)]^2 \quad (4.33)$$

onde $\hat{y}_i(t+j|t)$ é a predição ótima da i-ésima saída do sistema no instante de tempo $t+j$, N_{1i} e N_{2i} são, respectivamente, os horizontes mínimos e máximos de predição para a saída i, N_{ui} é o horizonte de controle da entrada i, $\lambda_i(j)$ e $\delta_i(j)$ são os pesos da i-ésima entrada e saída j instantes adiante, respectivamente, e $w_i(t+j)$ é a trajetória de referência futura para a i-ésima saída no instante $t+j$. Os horizontes totais de cada saída podem ser obtidos através da equação $N_i = N_{2i} - N_{1i} + 1$. A função custo dada na Eq. (4.33) é uma generalização da função utilizada no caso SISO.

A escolha dos horizontes, em geral, pelo fato de haver diferentes atrasos para cada saída do sistema, são feitas de forma independente para cada saída e entrada. Além disso, a normalização do modelo é muito importante para se obter uma ponderação adequada das variáveis. Se o modelo não estiver normalizado, os erros e as ações de controle na função custo não serão comparáveis em magnitude, assim, a escolha de λ_i e δ_i será mais difícil [41]. A normalização pode ser feita fazendo com que todas as variáveis tenham magnitude menor que 1, o que significa dividir cada variável pelo seu valor máximo esperado [42].

Assim como no caso SISO, um modelo CARIMA é utilizado para representar o sistema MIMO, com a diferença de que este modelo está no formato DMF:

$$\boldsymbol{A}(z^{-1})\boldsymbol{y}(t) = \boldsymbol{L}(z^{-1})\boldsymbol{B}(z^{-1})\boldsymbol{u}(t-1) + \boldsymbol{D}(z^{-1})\boldsymbol{v}(t) + \boldsymbol{T}(z^{-1})\frac{\boldsymbol{e}(t)}{\Delta} \quad (4.34)$$

onde estão presentes o vetor de saídas $\boldsymbol{y(t)} = [y_1(t) \dots y_n(t)]^T$, o vetor de entradas $\boldsymbol{u}(t-1) = [u_1(t-1) \dots u_m(t-1)]^T$, o vetor de perturbações $\boldsymbol{v(t)} = [v_1(t) \dots v_p]^T$, o vetor de ruídos brancos $\boldsymbol{e}(t) = [e_1(t) \dots e_n(t)]^T$ e $\Delta = 1 - z^{-1}$ que representa o operador diferença. $\boldsymbol{A}$ e $\boldsymbol{L}$ são matrizes diagonais $n \times n$ sendo que a primeira representa os denominadores das funções de transferência entre entradas e saídas, e a segunda representa o atraso mínimo com relação às entradas para cada saída. $\boldsymbol{B}$, de ordem $n \times m$, e $\boldsymbol{D}$, de ordem $n \times p$ representam os numeradores das funções

de transferência em relação às entradas e às perturbações, respectivamente. É importante ressaltar que os atrasos existentes com relação às perturbações ficam implícitos em $\boldsymbol{D}$. $\boldsymbol{T}$ é uma matriz diagonal $n \times n$ com o mesmo propósito do polinômio $T(z^{-1})$ no caso SISO, ou seja, pode ser utilizado para modelar características do ruído $e(t)$, ou como parâmetro de ajuste para melhorar a robustez do sistema em malha fechada.

Como a matriz $\boldsymbol{A}(z^{-1})$ é diagonal, é possível obter as predições ótimas de cada saída utilizando equações diofantinas independentes, assim, o seguinte modelo MISO é utilizado:

$$A_i(z^{-1})y_i(t) = z^{-d_i}\boldsymbol{B_i}(z^{-1})\boldsymbol{u}(t-1) + \boldsymbol{D_i}(z^{-1})\boldsymbol{v}(t) + \frac{T_i(z^{-1})}{\Delta}e_i(t) \tag{4.35}$$

onde $\boldsymbol{B}_i = [B_{i1}, \ldots, B_{im}]$, e $\boldsymbol{D}_i = [D_{i1}, \ldots, D_{ip}]$.

Executando um procedimento idêntico ao caso SISO, com uma única diferença no aumento no número de equações diofantinas necessárias devido à quantidade superior de entradas e perturbações, obtém-se a predição ótima da i-ésima saída:

$$\begin{aligned}\hat{y}_i(t+l|t) &= z^{-d_i}\boldsymbol{H_{il}}(z^{-1})\boldsymbol{\Delta u}(t-1+l) + \boldsymbol{I_{il}}(z^{-1})\frac{\boldsymbol{\Delta u}(t-1)}{T_i(z^{-1})} \\ &\quad + \boldsymbol{H_{v_{il}}}(z^{-1})\boldsymbol{\Delta v}(t+l) + \boldsymbol{I_{v_{il}}}(z^{-1})\frac{\boldsymbol{\Delta v}(t)}{T_i(z^{-1})} \\ &\quad + \frac{F_{il}(z^{-1})}{T_i(z^{-1})}y_i(t)\end{aligned} \tag{4.36}$$

onde $\boldsymbol{H_{il}}(z^{-1})$, $\boldsymbol{I_{il}}(z^{-1})$, $\boldsymbol{H_{v_{il}}}(z^{-1})$ e $\boldsymbol{I_{v_{il}}}(z^{-1})$ são vetores polinomiais onde os referentes às entradas possuem ordem $1 \times m$, e os referentes às saídas $1 \times p$. Estes vetores têm como elementos os polinômios resultantes das soluções das equações diofantinas para o i-ésima saída.

Repetindo este procedimento para as outras saídas, pode-se obter o vetor de predições futuras ótimas $\hat{\boldsymbol{y}}(t)$ através da seguinte equação:

$$\begin{aligned}\hat{\boldsymbol{y}}(t) &= \boldsymbol{H}\boldsymbol{u}(t) + \boldsymbol{H_v}(z^{-1})\boldsymbol{\Delta v}(t+1) + \boldsymbol{I}(z^{-1})\boldsymbol{\Delta u_f}(t-1) \\ &\quad + \boldsymbol{I_v}(z^{-1})\boldsymbol{\Delta v_f}(t) + \boldsymbol{F}(z^{-1})\boldsymbol{y_f}(t)\end{aligned} \tag{4.37}$$

onde

$$\hat{\boldsymbol{y}} = \begin{bmatrix} \hat{\boldsymbol{y_1}}(t) \\ \vdots \\ \hat{\boldsymbol{y_n}}(t) \end{bmatrix}, \qquad \hat{\boldsymbol{y_i}}(t) = \begin{bmatrix} \hat{y_i}(t+N_{1i}|t) \\ \vdots \\ \hat{y_i}(t+N_{2i}|t) \end{bmatrix},$$

$$\boldsymbol{u}(t) = \begin{bmatrix} \boldsymbol{\Delta u_1}(t) \\ \vdots \\ \boldsymbol{\Delta u_m}(t) \end{bmatrix}, \qquad \boldsymbol{\Delta u_i}(t) = \begin{bmatrix} \Delta u_i(t) \\ \vdots \\ \Delta u_i(t+N_{ui}-1) \end{bmatrix},$$

$$\boldsymbol{\Delta v_f}(t) = \begin{bmatrix} \boldsymbol{\Delta v_{f1}}(t) \\ \vdots \\ \boldsymbol{\Delta v_{fn}}(t) \end{bmatrix}, \qquad \boldsymbol{\Delta v_{fi}}(t) = \frac{1}{T_i(z^{-1})} \begin{bmatrix} \Delta v_1(t) \\ \vdots \\ \Delta v_p(t) \end{bmatrix},$$

$$\boldsymbol{\Delta u_f}(t-1) = \begin{bmatrix} \boldsymbol{\Delta u_{f1}}(t-1) \\ \vdots \\ \boldsymbol{\Delta u_{fn}}(t-1) \end{bmatrix}, \qquad \boldsymbol{\Delta u_{fi}}(t) = \frac{1}{T_i(z^{-1})} \begin{bmatrix} \Delta u_1(t) \\ \vdots \\ \Delta u_m(t) \end{bmatrix},$$

$$\boldsymbol{\Delta v}(t+1) = \begin{bmatrix} \Delta v_1(t+1) \\ \vdots \\ \Delta v_p(t+1) \end{bmatrix}, \qquad \boldsymbol{y_f}(t) = \begin{bmatrix} \frac{y_1(t)}{T_1(z^{-1})} \\ \vdots \\ \frac{y_n(t)}{T_i(z^{-1})} \end{bmatrix},$$

$$\boldsymbol{F}(z^{-1}) = \begin{bmatrix} \boldsymbol{F_1}(z^{-1}) & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{F_n}(z^{-1}) \end{bmatrix},$$

$$\boldsymbol{I}(z^{-1}) = \begin{bmatrix} \boldsymbol{I_1}(z^{-1}) & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{I_n}(z^{-1}) \end{bmatrix},$$

$$\boldsymbol{I_v}(z^{-1}) = \begin{bmatrix} \boldsymbol{I_{v1}}(z^{-1}) & & \boldsymbol{0} \\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{I_{vn}}(z^{-1}) \end{bmatrix}.$$

É importante notar que, caso $T_i(z^{-1}) = 1, \forall i$, não é preciso fa-

zer a filtragem de nenhuma das variáveis por $T_i(z^{-1})$. Desta forma, alguma das matrizes são alteradas: (i) $\boldsymbol{\Delta v_f}(t) = [\Delta v_1(t) \cdots \Delta v_p(t)]^T$, (ii) $\boldsymbol{\Delta u_f}(t-1) = [\Delta u_1(t-1) \cdots \Delta u_m(t-1)]^T$, (iii) $\boldsymbol{I}(z^{-1})$ e $\boldsymbol{I_v}(z^{-1})$ passam a ser vetores polinomiais $\boldsymbol{I}(z^{-1}) = [\boldsymbol{I_1}(z^{-1}) \cdots \boldsymbol{I_n}(z^{-1})]^T$, $\boldsymbol{I_v}(z^{-1}) = [\boldsymbol{I_{v1}}(z^{-1}) \cdots \boldsymbol{I_{vn}}(z^{-1})]^T$. Tem-se também que

$$\boldsymbol{H_v}(z^{-1}) = \begin{bmatrix} \boldsymbol{H_{v11}}(z^{-1}) & \cdots & \boldsymbol{H_{v1p}}(z^{-1}) \\ \vdots & \ddots & \vdots \\ \boldsymbol{H_{vn1}}(z^{-1}) & \cdots & \boldsymbol{H_{vnp}}(z^{-1}) \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H_{11}} & \cdots & \boldsymbol{H_{1m}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{H_{n1}} & \cdots & \boldsymbol{H_{nm}} \end{bmatrix}.$$

Da mesma forma que no caso SISO, a Eq. (4.37) pode ser simplificada agrupando as parcelas que não dependem das variações futuras da ação de controle na resposta livre $\boldsymbol{f}$:

$$\hat{\boldsymbol{y}} = \boldsymbol{Hu} + \boldsymbol{f}. \tag{4.38}$$

No caso sem restrições, a função custo da Eq. (4.33) pode ser reescrita da seguinte forma

$$J = (\boldsymbol{Hu} + \boldsymbol{f} - \boldsymbol{w})^T \boldsymbol{Q_y} (\boldsymbol{Hu} + \boldsymbol{f} - \boldsymbol{w}) + \boldsymbol{u}^T \boldsymbol{Q_u} \boldsymbol{u} \tag{4.39}$$

onde $\boldsymbol{Q_y} = diag(\boldsymbol{\delta_1}, \ldots, \boldsymbol{\delta_n})$ é uma matriz diagonal quadrada de ordem $\sum_{i=1}^{n} N_i$ que representa os pesos dos erros futuros e os pesos das ações de controle futuras são dadas por $\boldsymbol{Q_u} = diag(\boldsymbol{\lambda_1}, \ldots, \boldsymbol{\lambda_m})$, que também é diagonal quadrada, de ordem $\sum_{i=1}^{m} N_{ui}$. A solução da minimização de J é idêntica ao caso SISO, assim como o processo para obtê-la.

4.3.3 Exemplo e Implementação

Para ilustrar a aplicação do MIMO-GPC, e mostrar alguns detalhes da implementação, tem-se o seguinte sistema em tempo contínuo (onde as constante de tempo são dadas em minutos)

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{1}{0,7s+1} & \frac{5e^{-0,06s}}{0,3s+1} \\ \frac{1}{0,5s+1} & \frac{2}{0,4s+1} \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix} \tag{4.40}$$

Discretizando o modelo utilizando o sustentador de ordem zero, e considerando um período de amostragem de 0,03 minutos, tem-se o seguinte modelo discreto:

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} \frac{0,0420z^{-1}}{1-0,9580z^{-1}} & \frac{(0,4758z^{-1})z^{-2}}{1-0,9048z^{-1}} \\ \frac{0,0582z^{-1}}{1-0,9418z^{-1}} & \frac{0,1445z^{-1}}{1-0,9277z^{-1}} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \tag{4.41}$$

Considerando o modelo CARIMA da Eq. (4.34), as matrizes polinomiais do sistema, obtidos através da DMF são:

$$\boldsymbol{A}(z^{-1}) = \begin{bmatrix} 1 - 1,8629z^{-1} + 0,8669z^{-2} & 0 \\ 0 & 1 - 1,8695z^{-1} + 0,8737z^{-2} \end{bmatrix},$$

$$\boldsymbol{B}(z^{-1}) = \begin{bmatrix} 0,0420z^{-1} - 0,0380z^{-2} & (0,4758z^{-1} - 0,4559z^{-2})z^{-2} \\ 0,0582z^{-1} - 0,0540z^{-2} & 0,1445z^{-1} - 0,1361z^{-2} \end{bmatrix},$$

$$\boldsymbol{L}(z^{-1}) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Como o intuito deste exemplo é mostrar as matrizes do GPC, os horizontes de predição e controle foram escolhidos os menores possíveis, mantendo a estabilidade, de forma a facilitar a visualização. Os horizontes são $N_{11} = N_{12} = 1$, $N_{21} = N_{22} = 4$ e $N_{u1} = N_{u2} = 3$. As ponderações dos sinais de controle são iguais e constantes $\lambda_1(j) = \lambda_2(j) = 0,05, \forall j$, e as ponderações dos erros são iguais a 1, e o filtro $T_i(z^{-1}) = 1, \forall i$. Além disso, as referências futuras não são conhecidas. Com este ajuste, o tempo de acomodação do sistema é de aproximadamente 20 amostras ou 0,6 minutos.

A partir dos parâmetros de ajuste do controlador dado, é possível encontrar as matrizes polinomiais utilizadas para o cálculo das predições ótimas da Eq. (4.37) (obtidas através da solução de uma série de equações diofantinas):

$$\boldsymbol{H} = \left[\begin{array}{ccc|ccc} 0,042 & 0 & 0 & 0 & 0 & 0 \\ 0,0822 & 0,042 & 0 & 0 & 0 & 0 \\ 0,121 & 0,0822 & 0,042 & 0,476 & 0 & 0 \\ \hline 0,0582 & 0 & 0 & 0,144 & 0 & 0 \\ 0,113 & 0,0582 & 0 & 0,279 & 0,144 & 0 \\ 0,165 & 0,113 & 0,0582 & 0,403 & 0,279 & 0,144 \end{array}\right],$$

$$\boldsymbol{I}(z^{-1}) = \left[\begin{array}{c|c} -0,038 & 0,476z^{-1} - 0,456z^{-2} \\ -0,109 & 0,476 + 0,906z^{-1} - 1,3z^{-2} \\ -0,331 & 1,3 + 1,65z^{-1} - 2,49z^{-2} \\ \hline -0,054 & -0,136 \\ -0,155 & -0,391 \\ -0,296 & -0,747 \\ -0,473 & -1,19 \end{array}\right],$$

$$\boldsymbol{F}(z^{-1}) = \left[\begin{array}{c|c} 2,86 - 2,73z^{-1} + 0,867z^{-2} & 0 \\ 5.47 - 6,95z^{-1} + 2,48z^{-2} & 0 \\ 8,70 - 12,4z^{-1} + 4,74z^{-2} & 0 \\ 12,5 - 19,0z^{-1} + 7,54z^{-2} & 0 \\ \hline 0 & 2,87 - 2,74z^{-1} + 0,874z^{-2} \\ 0 & 5,49 - 7,0z^{-1} + 2,51z^{-2} \\ 0 & 8,76 - 12,6z^{-1} + 4,8z^{-2} \\ 0 & 12,6 - 19,2z^{-1} + 7,65z^{-2} \end{array}\right],$$

note que as matrizes foram subdivididas com linhas horizontais e verticais para evidenciar as sub-matrizes correspondentes às diferentes entradas e saídas. Como foi mostrado anteriormente, estas matrizes serão utilizadas para o cálculo da resposta livre do sistema e para obter a ação de controle ótima a ser aplicada no instante t.

Note que $\boldsymbol{I}(z^{-1})$ e $\boldsymbol{F}(z^{-1})$ são matrizes polinomiais. O modo mais fácil de representar este tipo de matriz em um programa de computador é utilizando matrizes normais, ou seja, seus elementos são números reais. Desta forma, cada coeficiente de um polinômio se torna um elemento de uma matriz vetor, por exemplo, o polinômio $F_{11}(z^{-1}) = 2,86 - 273z^{-1} + 0,867z^{-2}$ seria representado por $F_{11} = [2,86; -2,73; 0,867]$. Assim, operações com polinômios se tornam operações matriciais básicas, por exemplo

$$F_{11}(z^{-1})y(t) = [2,86; -2,73; 0,867] \begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \end{bmatrix}.$$

Utilizando esta representação em $\boldsymbol{I}(z^{-1})$ e $\boldsymbol{F}(z^{-1})$, têm-se que

$$\boldsymbol{I} = \left[\begin{array}{c|ccc} -0,038 & 0 & 0,476 & -0,456 \\ -0,109 & 0,476 & 0,906 & -1,3 \\ -0,331 & 1,3 & 1,65 & -2,49 \\ \hline -0,054 & -0,136 & 0 & 0 \\ -0,155 & -0,391 & 0 & 0 \\ -0,296 & -0,747 & 0 & 0 \\ -0,473 & -1,19 & 0 & 0 \end{array}\right],$$

$$\boldsymbol{F} = \left[\begin{array}{ccc|ccc} 2,86 & -2,73 & 0,867 & 0 & 0 & 0 \\ 5,47 & -6,95 & 2,48 & 0 & 0 & 0 \\ 8,70 & -12,4 & 4,74 & 0 & 0 & 0 \\ 12,5 & -19,0 & 7,54 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 2,87 & -2,74 & 0,874 \\ 0 & 0 & 0 & 5,49 & -7,0 & 2,51 \\ 0 & 0 & 0 & 8,76 & -12,6 & 4,8 \\ 0 & 0 & 0 & 12,6 & -19,2 & 7,65 \end{array}\right].$$

Nota-se que, na matriz $\boldsymbol{F}$, metade de seus elementos são nulos e que, na a matriz $\boldsymbol{I}$, existindo diferenças de ordem ou de atraso entre os polinômios $B_{ij}(z^{-1})$ para um j fixo, haverá colunas de zeros na matriz. Do ponto de vista prático, esta não é uma boa representação pois após cada período de amostragem, no cálculo da resposta livre, estas matrizes serão utilizadas em produtos matriciais e dezenas de multiplicações serão feitas onde uma de suas parcelas é zero, ou seja, não tem nenhum efeito no valor final da operação. Isto gera um desperdício de tempo computacional e de memória, recursos importantíssimos, principalmente em um sistema computacional onde estes são limitados.

Assim, para evitar esse problema, as sub-matrizes polinomiais de $\boldsymbol{F}(z^{-1})$ e de $\boldsymbol{I}(z^{-1})$, são armazenadas em memória separadamente. Desta forma, a resposta livre do sistema é calculada por partes da seguinte maneira:

$$\boldsymbol{f_i} = \sum_{j=1}^{m} \boldsymbol{I_{ij}}[\Delta u_j(t-1), \ldots, \Delta u_j(t-d_{ij}-nb_{ij})]^T + \boldsymbol{F_i}[y_i(t), \ldots, y_i(t-na_i)]^T \tag{4.42}$$

onde $\boldsymbol{f_i}$ é a resposta livre da i-ésima saída, d_{ij} é o atraso entre a entrada j e a saída i, e, nb_{ij} e na_i, são as ordens dos polinômios $B_{ij}(z^{-1})$ e $A_i(z^{-1})$, respectivamente. A resposta livre total é dada por

$$\boldsymbol{f} = \begin{bmatrix} \boldsymbol{f_1} \\ \vdots \\ \boldsymbol{f_n} \end{bmatrix}. \tag{4.43}$$

Calculando a resposta livre desta forma, elimina-se o problema descrito anteriormente de cálculos desnecessários durante os produtos matriciais, e do armazenamento em memória de elementos nulos que não têm efeito sobre valores finais das operações.

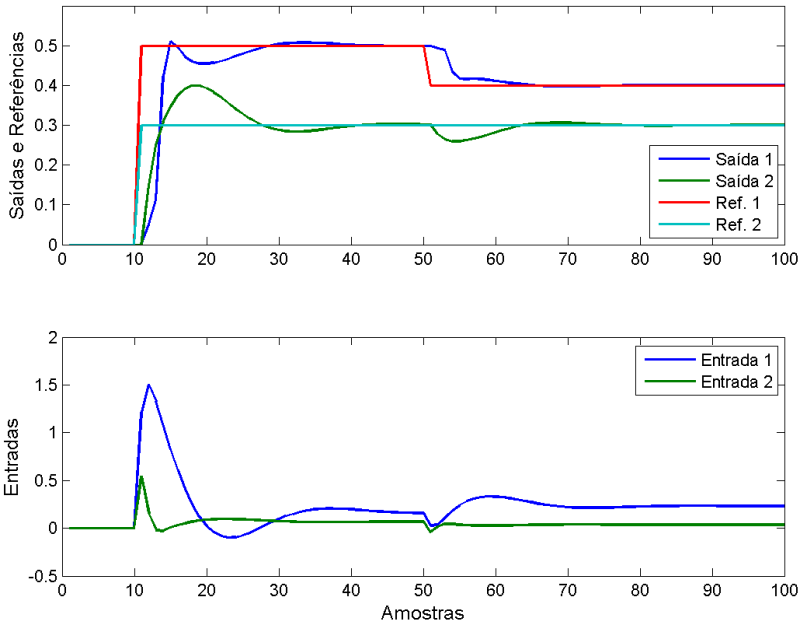


Figura 15 – Resultado da simulação do exemplo MIMO descrito.

Na Fig. (15) são mostrados os resultados da simulação do processo descrito sendo controlado pelo algoritmo MIMO-GPC com os parâmetros de controles já dados. Durante a simulação há uma mudança de referência de 0 para $0,5$ na Saída 1, e de 0 para $0,3$ na Saída 2 após 10 amostras. Existe uma segunda mudança de referência na Saída 1 para $0,4$ após 50 amostras. Observa-se que a resposta do sistema não apresenta um comportamento dominante de segunda ordem para a Saída 1, ao contrário do que normalmente se espera. Isto pode ser corrigido com o aumento dos horizontes de predição e controle, mas, neste caso, por simplicidade, optou-se por manter os horizontes em valores mínimos.

4.3.4 Algoritmo MIMO-GPC

Dado um processo MIMO, com m entradas, n saídas e p perturbações, representado por

$$\boldsymbol{Y}(z^{-1}) = \Gamma_u(z^{-1})\boldsymbol{U}(z^{-1}) + \Gamma_v(z^{-1})\boldsymbol{V}(z^{-1}), \tag{4.44}$$

onde $\boldsymbol{\Gamma_u}$ e $\boldsymbol{\Gamma_v}$ são matrizes de transferência discretas, o algoritmo MIMO-GPC é executado da seguinte forma:

- Passo 1: Calcular a representação em DMF do processo, obtendo as matrizes polinomiais $\boldsymbol{A}(z^{-1})$, $\boldsymbol{B}(z^{-1})$, $\boldsymbol{D}(z^{-1})$ e $\boldsymbol{L}(z^{-1})$

- Passo 2: Solucionar o conjunto de equações diofantinas dadas pelas equações (4.16), (4.22) e (4.23) e, considerando os horizontes de predição e controle, obter as matrizes $\boldsymbol{H}$, $\boldsymbol{H_v}$, $\boldsymbol{I_{ij}}$, $\boldsymbol{I_{v_{ik}}}$ e $\boldsymbol{F_i}$ dado que $i = 1, \ldots, n$, $j = 1, \ldots, m$ e $k = 1, \ldots, p$

- Passo 3: Leitura das saídas e perturbações do processo e, caso $T_i(z^{-1}) \neq 1$ para algum i, filtrar as variáveis de perturbação, entrada e saída pelo respectivo polinômio $T_i(z^{-1})$

- Passo 4: Cálculo da resposta livre por partes

  (a) Calcular a resposta livre da i-ésima saída através da equação

$$
\begin{aligned}
\boldsymbol{f_i} &= \sum_{j=1}^{m} \boldsymbol{I_{ij}}[\Delta u_{fj}(t-1), \ldots, \Delta u_{fj}(t-d_{ij}-nb_{ij})]^T \\
&\quad + \sum_{j=1}^{p} \boldsymbol{I_{v_{ij}}}[\Delta v_{fj}(t), \ldots, \Delta v_{fj}(t-d_{v_{ij}}-nd_{ij}+1)]^T \\
&\quad + \boldsymbol{F_i}[y_{fi}(t), \ldots, y_{fi}(t-na_i)]^T
\end{aligned}
$$

(b) Calcular a resposta livre total através da equação Eq. (4.43) e adicionar o termo $\boldsymbol{H_v \Delta v}(t+1)$ caso os valores das perturbações futuras sejam conhecidos

- Passo 5: Minimização da função custo
  (a) obter as matrizes $\boldsymbol{P}$, $\boldsymbol{q}^T$ e f_0 da Eq. (4.29)
  (b) Minimizar a função custo com o uso de um algoritmo de otimização quadrática e, assim, obter o vetor de incrementos das ações de controle futuras $\boldsymbol{u}$
- Passo 6: Calcular a ação a ser aplicada no instante atual dado que $u_j(t) = u_j(t-1) + \Delta u_j(t)$
- Passo 7: Aplicar a ação de controle, e esperar um tempo de amostragem
- Passo 8: Voltar ao Passo 3

Observe que o algoritmo anterior também se aplica a um processo SISO, que é um caso MIMO particular onde o número de entradas e saídas é igual a 1.

4.4 DTCGPC - CONTROLE PREDITIVO GENERALIZADO COM COMPENSAÇÃO DE ATRASO DE TRANSPORTE

Antes de descrever o algoritmo DTCGPC, é importante mostrar algumas das propriedades do GPC e o que levou ao desenvolvimento deste novo tipo de MPC. Será visto que o controlador GPC pode ser interpretado, no caso sem restrições, como um Preditor de Smith Filtrado (PSF) [41], mostrado na Fig. (16), ou seja, possui uma estrutura preditora interna (identificada na figura) de compensação do atraso dependente do filtro $F_r(z)$ e do modelo do processo, um controlador

primário $C(z)$ projetado para estabilização e conferir a dinâmica especificada, e um filtro de referência $F(z)$ que dá um grau de liberdade a mais no ajuste do controlador. Na Fig. (16), $P(z) = G(z)z^{-d}$ é a função de transferência real do processo, e $P_n(z) = G_n z^{-d_n}$ representa o modelo nominal utilizado no projeto do controlador.

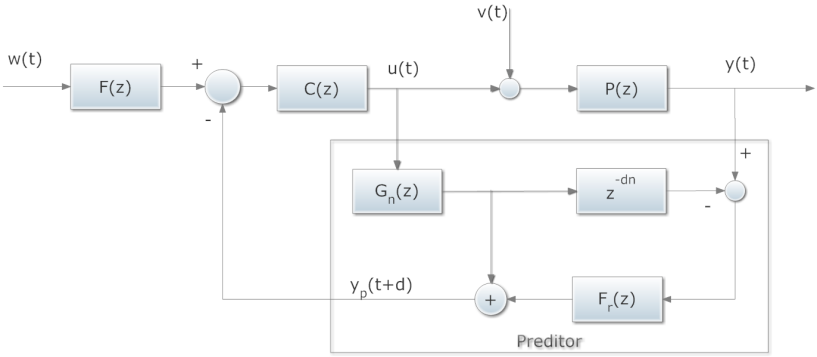


Figura 16 – Estrutura com dois graus de liberdade do Preditor de Smith Filtrado (PSF).

O controlador primário do PSF atua no sistema se baseando na predição do sistema para o tempo $t + d$ fornecida pela estrutura preditora. Desta forma o PSF consegue compensar internamente a presença do atraso. As predições são geradas a partir do modelo sem atraso do processo $G_n(z)$ e com um acréscimo de correção, dado pela diferença entre a saída real e a do modelo, chamado erro de predição. Este erro é filtrado por $F_r(z)$ que pode ser ajustado para acelerar a resposta do sistema na rejeição de perturbações, permitir o uso do PSF em plantas instáveis, e melhorar a robustez frente a erros de modelagem. Deve-se levar em conta, no entanto, que estas características são geralmente conflitantes, ou seja, melhorar a robustez implica em um tempo maior para rejeitar perturbações.

Para um sistema SISO descrito por

$$y(t) \quad = \quad z^{-d_n} \frac{B(z^{-1})}{A(z^{-1})} u(t-1)$$

é possível demonstrar que o controlador GPC, interpretado pela estrutura PSF, possui os seguintes blocos $C(z)$, $F(z)$ e $F_r(z)$:

$$C(z) = \frac{ly_1 + ly_2 z^{-1} + \ldots + ly_{na+1} z^{-na}}{(1 - z^{-1})(1 - lu_1 z^{-1} - lu_2 z^{-2} - \ldots - lu_{nb} z^{-nb})}, \quad (4.45)$$

$$F(z) = \frac{f_1 z^{d_n+1} + f_2 z^{d_n+2} + \ldots + f_N z^{d_n+N}}{ly_1 + ly_2 z^{-1} + \ldots + ly_{na+1} z^{-na}}, \quad (4.46)$$

$$F_r(z) = \frac{ly_1 F_{d_n}(z^{-1}) + ly_2 F_{d_n-1}(z^{-1}) + \ldots + ly_{na+1} F_{d_n-na}(z^{-1})}{ly_1 + ly_2 z^{-1} + \ldots + ly_{na+1} z^{-na}}. \quad (4.47)$$

onde $F_i(z^{-1})$ são polinômios provenientes das soluções das equações diofantinas, como já explicado na Seção 4.3, e os coeficiente ly_i, lu_i e f_i são dependentes dos polinômios $A(z^{-1})$ e $B(z^{-1})$, dos horizontes de predição e controle, e das ponderações $\delta(j)$ e $\lambda(j)$ [41]. Como é mostrado nas equações anteriores, o controlador primário no sistema equivalente do GPC tem a mesma ordem do modelo do processo, e o filtro de referência cancela os zeros de $C(z)$. Além disso, o filtro $F_r(z)$ utilizado para corrigir os erros entre a saída real e as predições depende dos parâmetros $A(z^{-1})$ e d_n do modelo, e dos parâmetros de ajuste λ e N.

A obtenção desta estrutura equivalente para o GPC foi um grande avanço pois permitiu uma análise mais detalhada da estrutura interna deste controlador. Esta estrutura evidencia o fato de que os algoritmos preditivos compensam intrinsecamente o atraso de transporte do processo e, através dela, é possível verificar como esta compensação é feita. A estrutura equivalente pode ser dividida em duas partes, o Preditor, que calcula as predições ótimas do sistema até $t + d_n$, e o Otimizador, que realiza o cálculo da ação de controle, a partir da referência futura e das predições do Preditor, através de um controlador ótimo $C(z)$. Esta ideia está ilustrada na Fig. (17) [41].

4.4.1 GPC e Robustez

Dada a estrutura do PSF na Fig. (16), a resposta do sistema de malha fechada para mudanças de referências, considerando o caso ideal ($P(z) = P_n(z)$), é dada por

$$H_r(z) = \frac{F(z)C(z)G_n(z)z^{-d_n}}{1 + C(z)G_n(z)}. \quad (4.48)$$

Para analisar a robustez do sistema em malha fechada, a planta

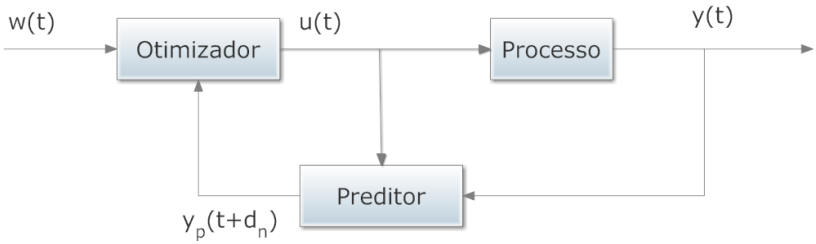


Figura 17 – Estrutura geral do GPC para processos com atraso de transporte.

é representada pela função de transferência $P(z)$

$$P(z) = P_n(z) + \Delta P(z) = P_n(z)[1 + \delta P(z)] \tag{4.49}$$

onde $\delta P(z)$ e $\Delta P(z)$ representam, respectivamente, os erros de modelagem na forma multiplicativa e aditiva. Para que o sistema em malha fechada seja considerado robusto, a seguinte condição deve ser satisfeita [41]

$$|\delta P(z)| < dP(z) = \frac{|1+C(z)G_n(z)|}{|C(z)G_n(z)F_r(z)|}, \quad z = e^{jw} \quad \forall \in [0, \pi]. \tag{4.50}$$

Assim, o índice de robustez do GPC pode ser obtido dado $H_r(z)$ da Eq. (4.48), e sabendo que $|e^{-jwd_n}| = 1$

$$|\delta P(z)| < dP(z) = \frac{|F(z)|}{|H_r(z)|}\frac{1}{|F_r(z)|}, \quad z = e^{jw} \quad \forall w \in [0, \pi]. \tag{4.51}$$

Esta equação mostra que, escolhidos os horizontes de predição e ponderações, a robustez do GPC é função do filtro $F_r(z)$, dado na Eq. (4.47). Como mostrado nesta equação, $F_r(z)$ é dependente do atraso nominal d_n, pois d_n define os elementos do numerador de $F_r(z)$ através dos polinômios $F_i(z^{-1})$ resultantes da solução das equações diofantinas. Por consequência, a robustez do controlador GPC é dependente do valor nominal do atraso. Considere, por exemplo, que o erro de modelagem é causado somente por um erro de estimação do atraso $\Delta d = d - d_n$. Assim

$$\delta P(w) = |e^{-jw\Delta d} - 1| \quad \forall w \in [0, \pi]. \tag{4.52}$$

Note que para um valor fixo de Δd, o erro de modelagem independe de d_n, enquanto que $dP(w)$ é função de d_n. Desta forma, o controlador GPC terá diferentes características de robustez para valores diferentes do atraso nominal.

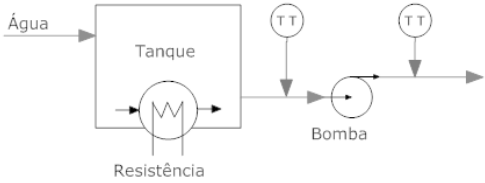


Figura 18 – Esquema do tanque de aquecimento de água.

Para ilustrar as consequências desta propriedade do GPC, tem-se o seguinte exemplo de um sistema de aquecimento de água, apresentado em [41], mostrado na Fig. (18). A função de transferência do sistema é

$$P_n(s) = \frac{e^{-Ls}}{2s+1} \tag{4.53}$$

onde o atraso de transporte L depende da posição do sensor de temperatura em relação à saída do tanque, e o tempo é medido em minutos. O atraso é $L = 0,4$ minutos se o sensor está mais próximo do tanque e $L = 4$ minutos se está na posição mais distante. Discretizando com um tempo de amostragem $T_s = 0,2$ min, a função de transferência discreta do sistema é $P_n(z) = \frac{0,095z^{-d_n}}{z-0,905}$, onde $d_n = 2$ para o primeiro caso e $d_n = 20$ no segundo caso. Os parâmetros do controlador são $\lambda = 1,5$ e $N = N_u = 15$. Deseja-se analisar a robustez do sistema em malha fechada frente a um erro de modelagem do atraso de ± 2 amostras, ou seja, o erro é representado pela Eq. (4.52) onde $\Delta d = 2$. Utilizando as equações (4.45) e (4.46) calculou-se os blocos $C(z)$ e $F(z)$ equivalentes do Otimizador do GPC:

$$C(z) = \frac{3,27(1-0,79z^{-1})}{1-z^{-1}}, \quad F(z) = \frac{0,21}{1-0,79z^{-1}}. \tag{4.54}$$

Nota-se que os blocos do Otimizador são iguais para os dois casos, já que este independe do atraso nominal do processo. Observa-se também que o numerador de $F(z)$ tem ordem zero, pois as referências futuras são consideradas iguais à do instante atual e o controlador as-

sume uma estrutura similar a um PI. Utilizando a Eq. (4.47), calcula-se o filtro $F_r(z)$ equivalente do GPC para os dois casos

$$F_{r1}(z) = \frac{1,22(1-0,827z^{-1})}{1-0,79z^{-1}}, \quad F_{r2}(z) = \frac{2,06(1-0,90z^{-1})}{1-0,79z^{-1}}. \quad (4.55)$$

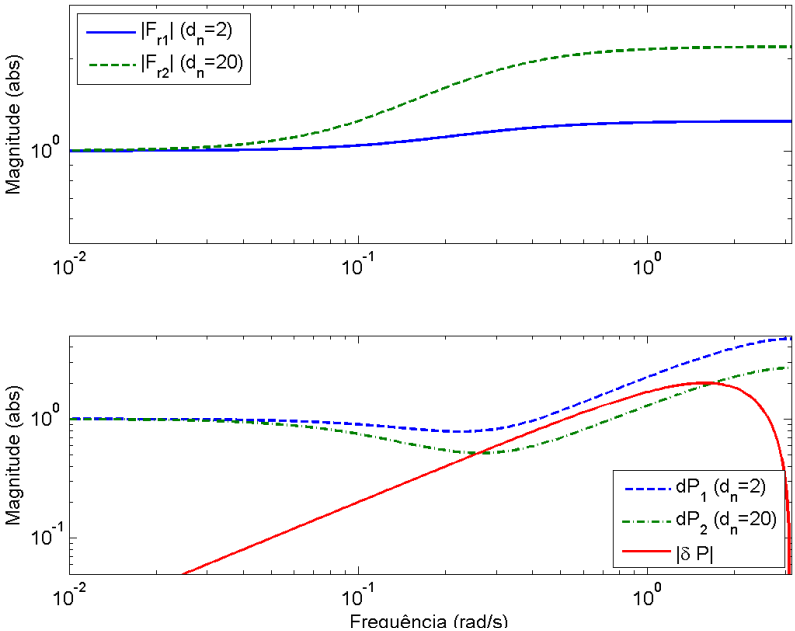


Figura 19 – Em cima, os diagramas de Magnitude dos filtros F_r e, embaixo, a comparação dos índices de robustez dP com a curva de magnitude do erro $|\delta P|$.

O primeiro gráfico na Fig. (19) mostra o diagrama de magnitude dos filtros $F_{ri}(z)$ calculados. Percebe-se que o aumento do atraso nominal, de $d_n = 2$ para $d_n = 20$, faz com que a característica passa-altas do filtro seja exarcebada. E, como mostrado pela Eq. (4.51), isto faz com que o sistema em malha fechada seja menos robusto a variações paramétricas. No segundo gráfico da Fig. (19) é feita a comparação do índice de robustez nos dois casos com o erro de modelagem. Fica evidente que no caso $d_n = 20$, não se pode garantir robustez para os erros paramétricos definidos. Na Fig. (20) são mostrados os resultados

de simulação para os casos $d_n = 2$ e $d_n = 20$, que são comparados com os resultados quando o atraso real é dado por, respectivamente, $d = 4$ e $d = 22$. No primeiro caso, com a adição do erro de modelagem, há uma deterioração da resposta, que fica mais oscilatória, mas o sistema ainda é estável. No caso $d_n = 20$, quando o atraso real é utilizado, o sistema fica instável, como era esperado devido aos fatos já mencionados.

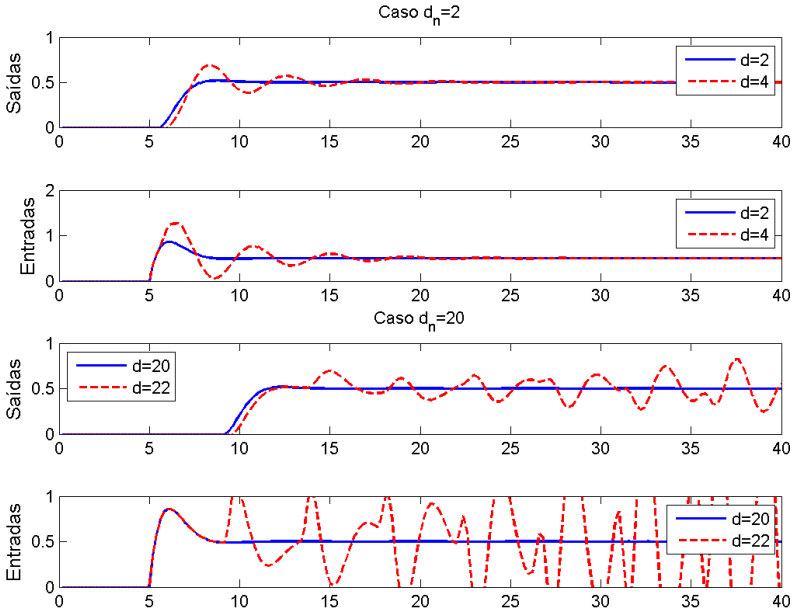


Figura 20 – Simulação do tanque de aquecimento de água para os casos $d_n = 2$ e $d_n = 20$, e comparação quando existe erro de modelagem no atraso de $\Delta d = \pm 2$ amostras e utilizando a sintonia apresentada.

Em [41, p. 311] foi mostrado que a condição

$$|F_r(e^{jw})| > 1 \quad \forall w \in [0, \pi] \tag{4.56}$$

pode ser comprovada em alguns casos particulares importantes, apesar de não ser possível provar analiticamente o caso geral. Na maioria dos casos é possível verificar que a condição $|F_r(e^{jw})| > 1$ para $w \in [w_0, \pi], w_0 > 0$, dado que w_0 é sempre menor que a frequência mínima onde os erros de modelagem devido ao atraso são importantes. Esta característica do GPC faz com que a estabilidade do sistema controlado

seja comprometida caso o atraso nominal seja grande.

4.4.1.1 Aumentando a Robustez do GPC com o Polinômio T

Na versão original do GPC, o polinômio T foi utilizado para representar as características estocásticas das perturbações. Depois notou-se que o uso deste polinômio alterava a robustez e a resposta a perturbações do sistema em malha fechada. No caso nominal, sem ruídos ou perturbações, as predições são as mesmas do que no caso com $T(z^{-1}) = 1$ e a ação de controle final $u(t)$ não depende de $T(z^{-1})$. Caso contrário, $T(z^{-1})$ atua como um filtro que pode ser utilizado para suprimir os efeitos de erros de modelagem em altas frequências. Neste caso, as predições não serão ótimas, mas robustez frente a variações paramétricas do modelo pode ser atingida [5, 43]. Esta propriedade foi utilizada por vários autores que propõem regras de ajuste para T onde este é utilizado para melhorar a robustez em vez de ser utilizado para modelar características estocásticas das perturbações, apesar de que seu ajuste não seja trivial [41].

Pode-se demonstrar que, considerando a estrutura PSF já descrita, $C(z)$ e $F(z)$ dependem das ponderações, dos horizontes e de $T(z^{-1})$. Além disso, o filtro $F_r(z^{-1})$ é dado por

$$F_r(z^{-1}) = \frac{1}{T(z^{-1})} \frac{ly_1 F_{d_n}(z^{-1}) + ly_2 F_{d_n-1}(z^{-1}) + \ldots + ly_{na+1} F_{d_n-na}(z^{-1})}{ly_1 + ly2z^{-1} + \ldots + ly_{na+1} z^{-na}}, \tag{4.57}$$

que mostra que $\frac{1}{T(z^{-1})}$ pode ser utilizado como um filtro passa-baixas para aumentar a robustez. Mas seu ajuste não é fácil pois os coeficientes ly_i, e os polinômios $F_i(z^{-1})$, dependem de $T(z^{-1})$.

Em [44], os autores sugerem que, para processo estáveis em malha aberta, a melhor escolha para obter uma melhora da robustez em altas frequências é

$$T(z^{-1}) = A(z^{-1})(1 - \beta z^{-1})^{N_1 - g} \tag{4.58}$$

onde β tem um valor próximo do pólo dominante de A, N_1 é o horizonte mínimo de predição e g é o grau do polinômio utilizado para gerar a referência futura do sistema.

Para exemplificar seu uso, retoma-se o exemplo mostrado na Fig. (18) onde, considerando um atraso nominal $d_n = 20$ e um erro de modelagem $\Delta d = 2$, o sistema se tornava instável para a sintonia

GPC apresentada. Mantendo-se o mesmos parâmetros horizontes e ponderações, adiciona-se o seguinte polinômio $T(z^{-1})$

$$T(z^{-1}) = (1 - 0,905z^{-1})(1 - 0,85z^{-1}) = 1 - 1,755z^{-1} + 0,7692z^{-2},$$

projetado utilizando a Eq. (4.58), onde se considera $N_1 - g = 1$. Na Fig. (21) estão os resultados de simulação onde se compara o caso sem e com o polinômio $T(z^{-1})$ projetado. Percebe-se que seu uso permitiu tornar o sistema mais robusto pois manteve a estabilidade do sistema em malha fechada.

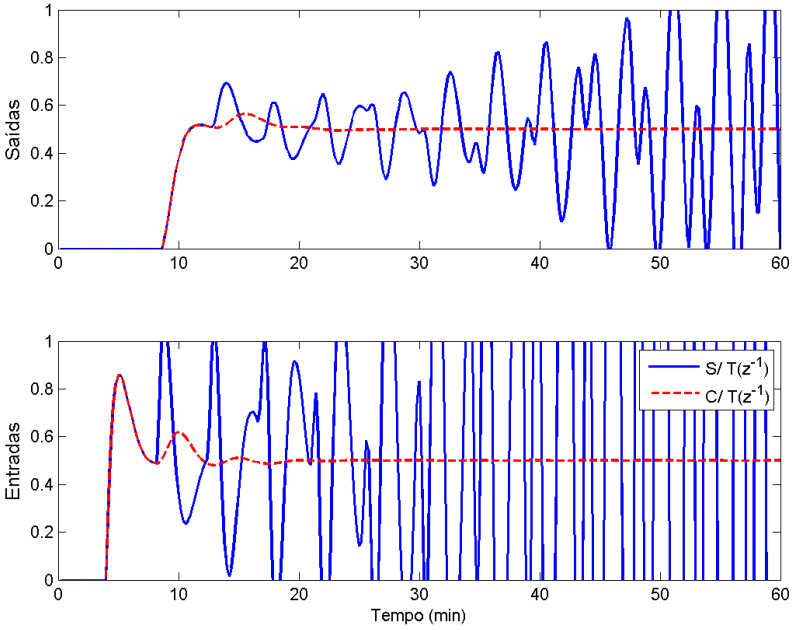


Figura 21 – Simulação do tanque de aquecimento de água considerando $d_n = 20$ e erro de modelagem $\Delta d = \pm 2$ nos casos com e sem o polinômio $T(z^{-1})$ projetado.

4.4.2 A Solução DTCGPC - Mudanças na Estrutura Preditora

Camacho e Normey-Rico mostram em [41, 45, 46] que, se fosse utilizado a estrutura preditora do PSF para calcular as predições até $t+d$, ao invés do preditor ótimo do GPC, e o Otimizador obtido através do procedimento normal do GPC, um controlador mais robusto e mais fácil de ajustar seria obtido.

Este controlador é o DTCGPC, que oferece várias vantagens se comparado ao GPC tradicional [41]: (i) tem a mesma performance nominal do GPC; (ii) o ajuste do filtro para melhorar a robustez é muito mais simples e oferece índices de robustez melhores do que o preditor ótimo do GPC com um filtro de mesma ordem; (iii) mantém o mesmo desempenho saída-referência independente da escolha do filtro.

4.4.2.1 Caso SISO

O procedimento para a obtenção do DTCGPC é essencialmente o mesmo do GPC tradicional, havendo apenas algumas alterações. Inicialmente, será visto como obter o Otimizador do DTCGPC, que calcula a ação de controle baseando-se nas predições até $t + d$, e depois como obter o novo Preditor baseado no PSF.

Dado um sistema SISO representado por

$$A(z^{-1})y(t) = B(z^{-1})z^{-d}u(t-1) + D(z^{-1})z^{-d_v}v(t) + \frac{e(t)}{\Delta}, \quad (4.59)$$

a predição do sistema no tempo $t + j$ é

$$\Delta A(z^{-1})y(t+j) = B(z^{-1})z^{-d}\Delta u(t-1+j)+D(z^{-1})z^{-d_v}\Delta v(t+j)+e(t+j). \quad (4.60)$$

Utilizando uma equação diofantina, é possível reescrever a equação anterior de tal forma que $y(t + j)$ dependa dos valores preditos até o tempo $t + d$. Essa equação é dada por

$$1 = E_j(z^{-1})\Delta A(z^{-1}) + z^{-l}F_j(z^{-1}), \quad (4.61)$$

onde $l = j - d$. Da Eq. (4.61) tem-se

$$\Delta A(z^{-1}) = \frac{1 - z^{-l}F_j(z^{-1})}{E_j}. \quad (4.62)$$

Substituindo a Eq. (4.62) na Eq. (4.60), obtém-se, após alguns rearranjos, a seguinte equação:

$$(1-z^{-l}F_j)y(t+j) = E_jB\Delta u(t-1-d+j)+E_jD\Delta v(t-d_v+j)+E_je(t+j). \tag{4.63}$$

Como o termo $E_je(t+j)$ só possui valores futuros do ruído, seu valor esperado futuro é zero. Assim, obtem-se a predição ótima do sistema em $t+j$, com $j > d$, dependente das predições até $t+d$:

$$\hat{y}(t+j|t) = E_jB\Delta u(t-1-d+j)+E_jD\Delta v(t-d_v+j)+F_jy(t+d). \tag{4.64}$$

A partir deste ponto, o procedimento é idêntico ao GPC. Utiliza-se as equações diofantinas

$$E_jB = H_j + z^{-k}I_j, \tag{4.65}$$
$$E_jD = H_{v_j} + z^{-k}I_{v_j}, \tag{4.66}$$

para evidenciar os termos dependentes dos incrementos futuros e passados de controle e de perturbação. Pode-se, da mesma forma do que no GPC, colocar as predições de $t+N_1$ até $t+N_2$ na forma matricial:

$$\begin{aligned}\hat{\boldsymbol{y}} &= \boldsymbol{H}\boldsymbol{u}(t) + \boldsymbol{H_v}(z^{-1})\Delta v(t+1) + \boldsymbol{I}(z^{-1})\Delta u(t-1) \\ &\quad +\boldsymbol{I_v}(z^{-1})\Delta v(t) + \boldsymbol{F}(z^{-1})\hat{y}(t+d).\end{aligned} \tag{4.67}$$

Tendo os valores preditos futuros de $y(t)$, pode-se utilizá-los na minimização da função custo J. Resta, então, apenas definir como serão calculadas as predições até $t+d$ pelo novo Preditor.

O Preditor ótimo do GPC será substituído pelo preditor do PSF, que é ilustrado na Fig. (22), onde $P_n(z^{-1}) = G_n(z^{-1})z^{-d} = \frac{z^{-d}B(z^{-1})}{A(z^{-1})}$ e $P_{vn} = G_{vn}(z^{-1})z^{-d_v} = \frac{D(z^{-1})z^{-d_v}}{A(z^{-1})}$, e $F_r(z^{-1})$ é o filtro de robustez já discutido.

A estrutura preditora destacada na Fig. (22) é utilizada somente para estudos do sistema e não pode ser utilizada diretamente, pois gera instabilidade interna no caso de processos instáveis em malha aberta. Assim, quando implementado, utiliza-se a estrutura da Fig. (23), ou seja,

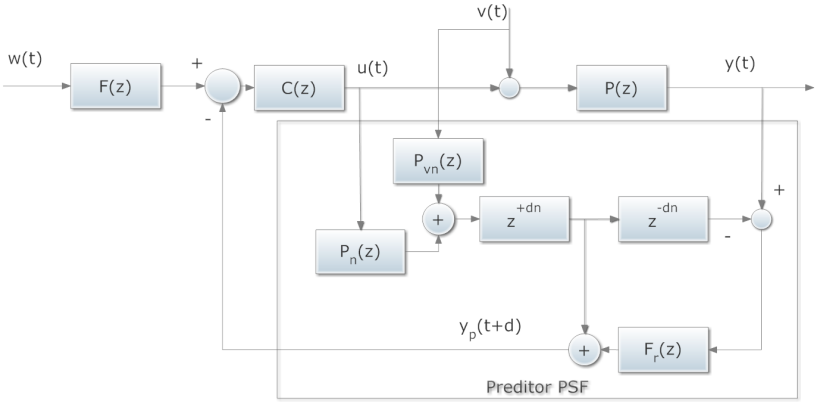


Figura 22 – Estrutura com dois graus de liberdade do PSF aplicado ao GPC.

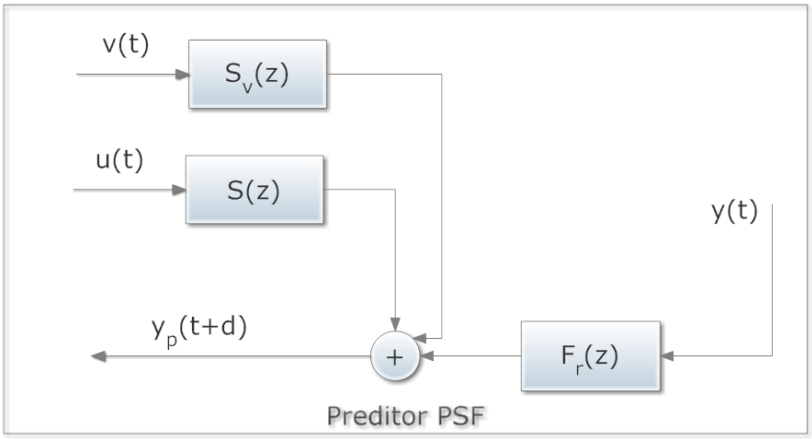


Figura 23 – Estrutura na forma de implementação do Preditor do PSF.

$$\hat{y}(t+d) = F_r(z^{-1})y(t) + S(z^{-1})u(t) + S_v(z^{-1})v(t), \tag{4.68}$$

onde $S = G_n - F_r P_n$ e $S_v = G_{vn} - F_r P_{vn}$, e F_r deve ser projetado de tal forma que S e S_v sejam estáveis. Caso $G_n(z^{-1})$ seja estável e não integrador, o ajuste do filtro $F_r(z^{-1})$ para aumentar a robustez é trivial, bastando escolher adequadamente a faixa de frequência onde

deverá haver atenuação devido aos possíveis erros de modelagem. Caso a planta seja instável, para cada pólo instável $|z_0| \geq 1$, o filtro $F_r(z^{-1})$ deve satisfazer as seguintes condições, como mostrado em [41, p. 328]:

$$\begin{cases} (F_r(z^{-1}) - z^d)|_{z=1} = 0 \\ (F_r(z^{-1}) - z^d)|_{z=z_0} = 0 \end{cases}, \tag{4.69}$$

e, no caso particular de uma planta integradora

$$\begin{cases} (F_r(z^{-1}) - z^d)|_{z=1} = 0 \\ \frac{d}{dz}(F_r(z^{-1}) - z^d)|_{z=z_0} = 0 \end{cases}. \tag{4.70}$$

Assim, tendo as predições até $t + d$ fornecidas pelo Preditor através da Eq. (4.68), o Otimizador é capaz de gerar as ações de controle que minimizarão a função custo J.

Antes de prosseguir para o caso MIMO, uma particularidade da função de transferência $S_v(z^{-1})$ deve ser discutida. Caso o atraso da perturbação d_v seja menor que o atraso nominal da planta d_n, $S_v(z^{-1})$ será não causal, ou seja, será dependente de valores futuros da perturbação v. Quando isto acontece, o algoritmo implementado considerará que os valores futuros da perturbação são iguais ao valor atual, ou seja, $v(t + k) = v(t)$, $\forall k \geq 1$.

4.4.2.2 Caso MIMO

Considerando um sistema MIMO com n saídas, m entradas e p perturbações, a obtenção das predições ótimas $\hat{y}_i(t + d|t)$ dependentes de $y_i(t + d)$ é feita seguindo o mesmo raciocínio por trás da extensão do caso SISO-GPC para MIMO-GPC. O sistema é representado pelo seguinte modelo

$$\hat{\boldsymbol{y}}(t) = \boldsymbol{\Gamma_u}(z^{-1})\boldsymbol{u}(t-1) + \boldsymbol{\Gamma_v}(z^{-1})\boldsymbol{v}(t), \tag{4.71}$$

onde $\boldsymbol{\Gamma_u}$ e $\boldsymbol{\Gamma_v}$ são matrizes de transferência de ordens $n \times m$ e $n \times p$, respectivamente, de tal forma que o elemento (i, j) é uma função de transferência discreta entre a j-ésima entrada (ou perturbação) e a i-ésima saída. A matriz de transferência pode ser decomposta em $\boldsymbol{\Gamma_u} = \boldsymbol{L}\overline{\boldsymbol{\Gamma_u}}$, onde $\boldsymbol{L}$ é uma matriz diagonal onde o i-ésimo elemento é o atraso mínimo entre as entradas e a i-ésima saída. Levando esse modelo em conta, a predição em $t + d$ é dada por

$$\hat{\boldsymbol{y}}(t+d) = \boldsymbol{F_r}(z^{-1})\boldsymbol{y}(t) + \boldsymbol{S}(z^{-1})\boldsymbol{u}(t) + \boldsymbol{S_v}(z^{-1})\boldsymbol{v}(t),$$

onde $\boldsymbol{F_r}(z^{-1})$ é uma matriz diagonal de ordem $n \times n$ onde o i-ésimo elemento é o polinômio $F_{ri}(z^{-1})$ ajustado de modo semelhante ao caso SISO, $\boldsymbol{S} = \overline{\boldsymbol{\Gamma_u}} - \boldsymbol{F_r}\boldsymbol{L}\boldsymbol{\Gamma_u}$ e $\boldsymbol{S_v} = \boldsymbol{L}^{-1}\boldsymbol{\Gamma_v} - \boldsymbol{F_r}\boldsymbol{\Gamma_v}$. Assim como no caso SISO, dependendo dos valores dos atrasos das perturbações e das entradas, $\boldsymbol{S_v}$ pode ser dependente de valores futuros das perturbações. Quando isto ocorrer, a solução adotada é a mesma do caso SISO, ou seja, considerar que os valores futuros sejam iguais ao valor atual da perturbação.

4.4.3 Algoritmo MIMO-DTCGPC

Dado um processo MIMO, com m entradas, n saídas e p perturbações, representado por

$$\boldsymbol{Y}(z^{-1}) = \boldsymbol{\Gamma_u}(z^{-1})\boldsymbol{U}(z^{-1}) + \boldsymbol{\Gamma_v}(z^{-1})\boldsymbol{V}(z^{-1}), \tag{4.72}$$

onde Γ_1 e Γ_2 são matrizes de transferência discretas, o algoritmo MIMO-DTCGPC é executado da seguinte forma:

- Passo 1: Calcular a representação em DMF do processo, obtendo as matrizes polinomiais $\boldsymbol{A}(z^{-1})$, $\boldsymbol{B}(z^{-1})$, $\boldsymbol{D}(z^{-1})$ e $\boldsymbol{L}(z^{-1})$
- Passo 2: Solucionar o conjunto de equações diofantinas dadas pelas equações (4.61), (4.65) e (4.66) considerando os horizontes de predição e controle, obtendo as matrizes $\boldsymbol{H}$, $\boldsymbol{H_v}$, $\boldsymbol{I_{ij}}$, $\boldsymbol{I_{v_{ik}}}$ e $\boldsymbol{F_i}$ dado que $i = 1, \ldots, n$, $j = 1, \ldots, m$ e $k = 1, \ldots, p$
- Passo 3: Cálculo do Preditor baseado no PSF, dado $\boldsymbol{F_r}(z^{-1})$ e o modelo por matriz de transferência discreta da Eq. (4.72), obtendo as matrizes discretas $\boldsymbol{S}(z^{-1})$ e $\boldsymbol{S_q}(z^{-1})$
- Passo 4: Leitura das saídas e perturbações do processo
- Passo 5: Obtenção das predições até o tempo $t + d$ através do Preditor
- Passo 6: Cálculo da resposta livre por partes
  (a) Calcular a resposta livre da i-ésima saída através da equação

$$\begin{aligned}\boldsymbol{f_i} &= \sum_{j=1}^{m} \boldsymbol{I_{ij}}[\Delta u_j(t-1), \ldots, \Delta u_j(t-d_{ij}-nb_{ij})]^T \\ &+ \sum_{j=1}^{p} \boldsymbol{I_{v_{ij}}}[\Delta v_j(t), \ldots, \Delta v_j(t-d_{v_{ij}}-nd_{ij}+1)]^T \\ &+ \boldsymbol{F_i}[\hat{y}_i(t+d), \ldots, \hat{y}_i(t-na_i)]^T\end{aligned}$$

(b) Calcular a resposta livre total através da Eq. (4.43) e adicionar o termo $\boldsymbol{H_v}\boldsymbol{\Delta v}(t+1)$ caso os valores das perturbações futuras sejam conhecidos

- Passo 7: Minimização da função custo
  (a) obter as matrizes $\boldsymbol{P}$, $\boldsymbol{q}^T$ e f_0 da Eq. (4.29)
  (b) Minimizar a função custo com o uso de um algoritmo de otimização quadrática e assim obter o vetor de incrementos das ações de controle futuras $\boldsymbol{u}$
- Passo 8: Calcular a ação a ser aplicada no instante atual dado que $u_j(t) = u_j(t-1) + \Delta u(t)$
- Passo 9: Aplicar a ação de controle e esperar um tempo de amostragem
- Passo 10: Voltar ao Passo 4

4.5 SSMPC - CONTROLE PREDITIVO POR ESPAÇO DE ESTADOS

Dado um processo multivariável com m entradas, n saídas e l estados descrito pela seguinte modelo por espaço de estados

$$\begin{aligned}\boldsymbol{x}(t+1) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{\Delta u}(t) + \boldsymbol{P}\boldsymbol{v}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{e}(t)\end{aligned} \tag{4.73}$$

onde $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{P}$ são matrizes de ordem $l \times l$, $l \times m$, $n \times l$ e $l \times 1$, respectivamente. $\boldsymbol{v}(t)$ e $\boldsymbol{e}(t)$ são vetores de ruídos brancos de média nula que afetam, respectivamente, os estados e as saídas. As saídas do processo

no instante $t+j$ podem ser calculados recursivamente aplicando a Eq. (4.73), resultando em:

$$\begin{aligned} \boldsymbol{y}(t+j) &= \boldsymbol{C}\boldsymbol{A}^{j}\boldsymbol{x}(t) + \sum_{i=0}^{j-1} \boldsymbol{C}\boldsymbol{A}^{j-i-1}\boldsymbol{B}\boldsymbol{\Delta}u(t+i) + \\ &\quad + \sum_{i=0}^{j-1} \boldsymbol{C}\boldsymbol{A}^{j-i-1}\boldsymbol{P}v(t+i) + e(k+j) \end{aligned} \tag{4.74}$$

Considerando que o valor esperado para os ruídos no futuro é zero, a predição ótima j passos à frente do instante t é dada por:

$$\hat{\boldsymbol{y}}(t+j|t) = \boldsymbol{C}\boldsymbol{A}^{j}\boldsymbol{x}(t) + \sum_{i=0}^{j-1} \boldsymbol{C}\boldsymbol{A}^{j-i-1}\boldsymbol{B}\boldsymbol{\Delta}\boldsymbol{u}(t+i) \tag{4.75}$$

Agora, calculando todos os valores futuros ótimos da saída considerando um horizonte de predição de N_2 e um horizonte de controle de $N_u = N_2$, tem-se:

$$\begin{aligned} \boldsymbol{y} = \begin{bmatrix} \hat{\boldsymbol{y}}(t+1|t) \\ \hat{\boldsymbol{y}}(t+2|t) \\ \vdots \\ \hat{\boldsymbol{y}}(t+N_2|t) \end{bmatrix} &= \begin{bmatrix} \boldsymbol{C}\boldsymbol{A} \\ \boldsymbol{C}\boldsymbol{A}^{2} \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^{N_2} \end{bmatrix} \boldsymbol{x}(t) \\ &\quad + \begin{bmatrix} \boldsymbol{C}\boldsymbol{B}\boldsymbol{\Delta}\boldsymbol{u}(t) \\ \sum_{i=0}^{1} \boldsymbol{C}\boldsymbol{A}^{1-i}\boldsymbol{B}\boldsymbol{\Delta}\boldsymbol{u}(t+i) \\ \vdots \\ \sum_{i=0}^{N_2-1} \boldsymbol{C}\boldsymbol{A}^{N_2-1-i}\boldsymbol{B}\boldsymbol{\Delta}\boldsymbol{u}(t+i) \end{bmatrix} \end{aligned} \tag{4.76}$$

que pode ser expresso como

$$\boldsymbol{y} = \boldsymbol{M}x(t) + \boldsymbol{H}\boldsymbol{u} \tag{4.77}$$

onde $\boldsymbol{H}$ é uma matriz quadrada triangular inferior onde seus elementos são dados por $\boldsymbol{H}_{ij} = \boldsymbol{C}\boldsymbol{A}^{i-j}\boldsymbol{B}$ e $\boldsymbol{u} = [\boldsymbol{\Delta}\boldsymbol{u}(t)^T, \ldots, \boldsymbol{\Delta}\boldsymbol{u}(t+N_2-1)^T]^T$. É preciso ressaltar que o cálculo das predições depende da leitura dos estados $\boldsymbol{x}(t)$. Como em muitos casos não é possível obter diretamente

o valor destas variáveis, utiliza-se um observador de estados baseado no filtro de Kalman para obter uma estimação de $\boldsymbol{x}(t)$ [5].

Considerando o conjunto de predições das quais a função objetivo depende, ou seja, entre $t + N_1$ e $t + N_2$: $\boldsymbol{y}_{N_{12}} = [\hat{\boldsymbol{y}}(t + N_1|t)^T, \ldots, \hat{\boldsymbol{y}}(t + N_2|t)^T]^T$ e o vetor com as N_3 ações de controle futuras $\boldsymbol{u} = [\boldsymbol{\Delta u}(t)^T, \ldots, \boldsymbol{\Delta u}(t + N_3 - 1)^T]^T$. Tem-se

$$\boldsymbol{y_{N_{12}}} = \boldsymbol{M_{N_{12}}}\boldsymbol{x}(t) + \boldsymbol{H_{N_{123}}}\boldsymbol{u_{N_3}} \tag{4.78}$$

onde as matrizes $\boldsymbol{M_{N_{12}}}$ e $\boldsymbol{H_{N_{123}}}$ são formadas pelas correspondentes sub-matrizes em $\boldsymbol{M}$ e $\boldsymbol{H}$, respectivamente. É importante observar que a resposta livre do sistema é dada por $\boldsymbol{f} = \boldsymbol{M_{N_{12}}}x(t)$.

A função custo Eq. (4.33) pode ser reescrita como:

$$\begin{aligned} J \;\; = \;\; & (\boldsymbol{H_{N_{123}}}\boldsymbol{u_{N_3}} + \boldsymbol{M_{N_{12}}}\boldsymbol{x} - \boldsymbol{w})^T \boldsymbol{Q_y}(\boldsymbol{H_{N_{123}}}\boldsymbol{u_{N_3}} + \boldsymbol{M_{N_{12}}}\boldsymbol{x} - \boldsymbol{w}) \\ & + \boldsymbol{u_{N_3}}^T \boldsymbol{Q_u}\boldsymbol{u_{N_3}} \end{aligned} \tag{4.79}$$

No caso sem restrições, a solução ótima é dada por:

$$\boldsymbol{u} = (\boldsymbol{H}_{\boldsymbol{N_{123}}}^{\boldsymbol{T}}\boldsymbol{Q_y}\boldsymbol{H_{N_{123}}} + \boldsymbol{Q_u})^{-1}\boldsymbol{H_{N_{123}}}\boldsymbol{Q_y}(\boldsymbol{w} - \boldsymbol{M_{N_{12}}}\boldsymbol{x}(t)) \tag{4.80}$$

4.5.1 Algoritmo SSMPC

Dado um processo MIMO, com m entradas, n saídas e l estados, representado pela Eq. (4.73), o algoritmo SSMPC é executado da seguinte forma:

- Passo 1: Obter as matrizes $\boldsymbol{M}$ e $\boldsymbol{H}$ da Eq. (4.77), e calcular $\boldsymbol{F_{N_{12}}}$ e $\boldsymbol{H_{N_{123}}}$ de acordo com os horizontes de predição e controle
- Passo 2: Leitura dos estados do processo $\boldsymbol{x}(t)$. Utilizar um observador de estados caso nem todos sejam mensuráveis
- Passo 3: Cálculo da resposta livre através da equação

$$\boldsymbol{f} = \boldsymbol{M_{N_{12}}}x(t)$$

- Passo 4: Minimização da função custo
  (a) obter as matrizes $\boldsymbol{P}$, $\boldsymbol{q}^T$ e f_0 da Eq. (4.29)

(b) Minimizar a função custo com o uso de um algoritmo de otimização quadrática e assim obter o vetor de incrementos das ações de controle futuras $\boldsymbol{u}$

- Passo 5: Calcular a ação a ser aplicada no instante atual dado que $\boldsymbol{u}(t) = \boldsymbol{u}(t-1) + \boldsymbol{\Delta u}(t)$

- Passo 6: Aplicar a ação de controle e esperar um tempo de amostragem

- Passo 7: Voltar ao Passo 2

4.6 MPC COM RESTRIÇÕES

Nenhum dos algoritmos MPC descritos nas seções anteriores levam em conta o fato de que as variáveis de um processo são limitadas em amplitude e estão sujeitas a vários tipos de restrições, o que não é uma situação realística. Sistemas de atuação são limitados em amplitude e na velocidade com que mudam de valor, por exemplo, uma válvula tem sua abertura limitada de 0 a 100 %, assim como a velocidade com que ela pode abrir ou fechar. As variáveis de processo também são limitadas e estes limites podem estar relacionados com aspectos físicos, por exemplo a altura máxima de um tanque, de segurança, a violação de restrições pode causar danos a equipamentos e à vida de pessoas, e de produção, certos tipos de processos necessitam que as variáveis estejam dentro de determinadas faixas de valores para garantir a qualidade do produto sendo produzido [5].

Controladores clássicos como o PID têm dificuldades em lidar com restrições, especialmente aquelas ligadas às variáveis controladas, pois estes controladores não "sabem" como o sistema se comportará no futuro, assim, não podem corrigir a ação de controle para evitar a violação das restrições, ao contrário do MPC. O algoritmo MPC faz uma predição do comportamento futuro do sistema, desta forma é possível calcular a ação de controle de tal maneira que as restrições na variável predita não sejam violadas [47].

As restrições mais comuns agindo em um processo podem existir devido a limites de amplitude no sinal de controle, taxa de variação máxima no atuador e limites nas saídas. Para um sistema MIMO com m entradas e n saídas, estas restrições podem ser descritas matematicamente como [5]:

$$\begin{array}{rcccll} U_{i,min} & \leq & u_i(t) & \leq & U_{i,max}, & \forall t, i = 1, \ldots, m \\ \Delta u_{i,min} & \leq & \Delta u_i(t) & \leq & \Delta u_{i,max}, & \forall t, i = 1, \ldots, m \\ y_{i,min} & \leq & y_i(t) & \leq & y_{i,max}, & \forall t, i = 1, \ldots, n \end{array} \tag{4.81}$$

Todas estas restrições podem ser escritas de forma matricial:

$$\begin{array}{rcccl} \mathbf{1}_{\boldsymbol{N_{ui}}} U_{i,min} & \leq & \boldsymbol{T}_{\boldsymbol{N_{ui}}} \boldsymbol{\Delta u_i}(t) + \mathbf{1}_{\boldsymbol{N_{ui}}} u_i(t-1) & \leq & \mathbf{1}_{\boldsymbol{N_{ui}}} U_{i,max} \\ \mathbf{1}_{\boldsymbol{N_{ui}}} \Delta u_{i,min} & \leq & \boldsymbol{\Delta u_i}(t) & \leq & \mathbf{1}_{\boldsymbol{N_{ui}}} \Delta u_{i,max} \\ \mathbf{1}_{\boldsymbol{N_i}} y_{i,min} & \leq & \boldsymbol{\hat{y}_i}(t) & \leq & \mathbf{1}_{\boldsymbol{N_i}} y_{i,max} \end{array} \tag{4.82}$$

onde $\mathbf{1}_{\boldsymbol{x}}$ é um vetor coluna com x elementos iguais a 1 e $\boldsymbol{T}_{\boldsymbol{x}}$ é uma matriz quadrada triangular inferior de ordem x cujos elementos não nulos são iguais a 1. Sabendo que $\hat{\boldsymbol{y}}(t) = \boldsymbol{H}\boldsymbol{u} + \boldsymbol{f}$, e estendendo os princípios utilizados na Eq. (4.81) para o restante das entradas e saídas, pode-se representar as restrições dadas da seguinte forma:

$$\begin{bmatrix} \boldsymbol{R_U} \\ \boldsymbol{R_{\Delta u}} \\ \boldsymbol{R_y} \end{bmatrix} \boldsymbol{u} \leq \begin{bmatrix} \boldsymbol{c_U} \\ \boldsymbol{c_{\Delta u}} \\ \boldsymbol{c_y} \end{bmatrix} \tag{4.83}$$

onde $\boldsymbol{R_x}$ e $\boldsymbol{c_x}$ são matrizes que representam as restrições em x. Os valores destas matrizes são dados a seguir:

$$\boldsymbol{R_y} = \begin{bmatrix} \boldsymbol{H} \\ \hline \boldsymbol{-H} \end{bmatrix}, \; \boldsymbol{c_y} = \begin{bmatrix} \mathbf{1}_{\boldsymbol{N_1}} y_{1,max} \\ \vdots \\ \mathbf{1}_{\boldsymbol{N_n}} y_{n,max} \\ \hline -\mathbf{1}_{\boldsymbol{N_1}} y_{1,min} \\ \vdots \\ -\mathbf{1}_{\boldsymbol{N_n}} y_{n,min} \end{bmatrix} + \begin{bmatrix} \boldsymbol{-f} \\ \hline \boldsymbol{f} \end{bmatrix}$$

$$\boldsymbol{R_{\Delta u}} = \begin{bmatrix} \boldsymbol{I}_{N_u} \\ \hline \\ -\boldsymbol{I}_{N_u} \end{bmatrix}, \; \boldsymbol{c_{\Delta u}} = \begin{bmatrix} \mathbf{1}_{\boldsymbol{N_{u1}}} \Delta u_{1,max} \\ \vdots \\ \mathbf{1}_{\boldsymbol{N_{um}}} \Delta u_{m,max} \\ \hline -\mathbf{1}_{\boldsymbol{N_{u1}}} \Delta u_{1,min} \\ \vdots \\ -\mathbf{1}_{\boldsymbol{N_{um}}} \Delta u_{m,min} \end{bmatrix}$$

$$\boldsymbol{R_U} = \left[\begin{array}{ccc} \boldsymbol{T_{N_{u1}}} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \boldsymbol{T_{N_{um}}} \\ \hline -\boldsymbol{T_{N_{u1}}} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & -\boldsymbol{T_{N_{um}}} \end{array}\right],$$

$$\boldsymbol{c_U} = \left[\begin{array}{c} \mathbf{1}_{\boldsymbol{N_{u1}}} U_{1,max} - \mathbf{1}_{\boldsymbol{N_{u1}}} u_1(t-1) \\ \vdots \\ \mathbf{1}_{\boldsymbol{N_{um}}} U_{m,max} - \mathbf{1}_{\boldsymbol{N_{um}}} u_m(t-1) \\ \hline -\mathbf{1}_{\boldsymbol{N_{u1}}} U_{1,min} + \mathbf{1}_{\boldsymbol{N_{u1}}} u_1(t-1) \\ \vdots \\ -\mathbf{1}_{\boldsymbol{N_{um}}} U_{m,min} + \mathbf{1}_{\boldsymbol{N_{um}}} u_m(t-1) \end{array}\right],$$

onde $\boldsymbol{I_{N_u}}$ é uma matrix identidade de ordem $\sum_{i=1}^{m} N_{ui}$.

Desta maneira, o cálculo da ação de controle é dado pelo seguinte problema de otimização quadrático:

$$\begin{array}{ll} \text{Minimizar:} & J(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T \boldsymbol{P}\boldsymbol{u} + \boldsymbol{q}^T\boldsymbol{u} + f_0 \\ \text{Sujeito a:} & \overline{\boldsymbol{R}}\boldsymbol{u} \leq \overline{\boldsymbol{c}} \end{array} \tag{4.84}$$

onde a matriz $\overline{\boldsymbol{R}}$ e o vetor $\overline{\boldsymbol{c}}$ são obtidos combinando as matrizes $\boldsymbol{R_x}$ e $\boldsymbol{c_x}$ dadas anteriormente.

Ao contrário do caso sem restrições, o problema da Eq. (4.84) não tem solução algébrica, assim, é necessário utilizar algoritmos de otimização quadrática tais como, por exemplo, os algoritmos de Conjunto Ativo (*Active Set Methods*) ou de Ponto Interior (*Interior-Point Methods*).

Quando se considera restrições, a parte de otimização dos algoritmos MPC descritos anteriormente devem ser ligeiramente modificados:

- Passo: Minimização da função custo
  (a) obter as matrizes $\boldsymbol{P}$, $\boldsymbol{q}^T$ e f_0 da Eq. (4.29)
  (b) calcular as matrizes $\overline{\boldsymbol{R}}$ e $\overline{\boldsymbol{c}}$ considerando as restrições dadas
  (c) Minimizar a função custo com o uso de um algoritmo de otimização quadrática e assim obter o vetor de incrementos das ações de controle futuras $\boldsymbol{u}$

Existem outros tipos de restrições que podem ser consideradas,

como por exemplo, não permitir sobressinal nos transitórios das respostas ou mesmo restrições para lidar com dinâmicas de fase não-mínima. Mais detalhes podem ser encontrados em [5].

4.6.1 Controle por Faixas da Saída

Um modo de operação interessante para a indústria, apesar de não ser uma restrição do sistema, é o controle por faixas da saída, ou seja, a saída não precisa seguir um valor específico de referência, desta forma, ela pode variar dentro de uma faixa pré-definida

$$yf_{min} \leq y(t) \leq yf_{max}, \forall t.$$

Consegue-se este tipo de operação da seguinte forma [48]:

- Para cada saída i, é observada sua predição no instante $t + j$
- Se $yf_{i\,min} \leq \hat{y}_i(t+j|t) \leq yf_{i\,max}$, a saída $\hat{y}_i(t+j|t)$ deve ser ignorada na função custo, pois ela já está dentro da faixa especificada. Isto é feito modificando o seu peso na matriz de ponderação dos erros futuros para 0.
- Se $\hat{y}_i(t+j|t) > yf_{i\,max}$, a saída y_i deve ser trazida para seu limite superior. Portanto, faz-se $w_i(t + j) = yf_{i\,max}$, e é utilizado a ponderação especificada no ajuste do controlador.
- Se $\hat{y}_i(t+j|t) < yf_{i\,min}$, a saída y_i deve ser trazida para seu limite inferior. Portanto, faz-se $w_i(t + j) = yf_{i\,min}$, e é utilizado a ponderação especificada no ajuste do controlador.

4.6.2 Particularidades do Uso de Restrições

O uso de restrições em conjunto com os algoritmos MPC traz diversas vantagens. Por exemplo, em certos processos, a violação de restrições leva a paradas de emergência da planta, que evitam danos aos equipamentos e riscos de acidentes com empregados. Como é possível, com o controle preditivo, levar estas restrições em conta no cálculo das ações de controle, diminuí-se consideravelmente a chance de paradas de emergências e, por consequência, o tempo não produtivo da planta.

No entanto, o uso de restrições aumenta a complexidade do cálculo da ação de controle, pois não é mais possível encontrar uma solução algébrica para o problema quadrático MPC, sendo necessária

a utilização de algoritmos de otimização. O aumento da complexidade do cálculo se traduz em um gasto de tempo maior para gerar o valor de controle atual, assim, é preciso saber de antemão qual o tempo que o sistema embarcado gasta com esta operação, pois, se o problema for complexo demais, o tempo necessário será maior que o período de amostragem especificado, inviabilizando a utilização do sistema de controle proposto.

Apesar de haver uma correlação entre o tempo para solucionar o problema quadrático, o número de variáveis de decisão (que é igual à soma dos horizontes de controle) e a quantidade de restrições, não é possível saber *a priori* o tempo necessário para calcular a ação de controle com o algoritmo de otimização utilizado, fornecido pela biblioteca Python CVXOPT. Assim, é necessário fazer um análise através de simulações do problema para verificar a viabilidade do uso em determinada planta, como será mostrado no capítulo 6. Ainda assim, pode ocorrer de, em uma situação real, o tempo de cálculo ultrapassar o período de amostragem. Nestes casos, o programa do sistema embarcado muda o modo de controle para manual e aciona um alarme.

Em [49], foi mostrado que é possível utilizar uma solução subótima para o problema quadrático imposto pelos algoritmos MPC e ainda assim garantir a estabilidade do processo controlado. Assim, futuramente, será implementada uma biblioteca de otimização própria que faça uso dessa propriedade, ou seja, que detecte se o período de amostragem vai ser extrapolado e que, verificada esta condição, interrompa a otimização e forneça uma solução sub-ótima para ser aplicada.

Um outro problema que surge ao se utilizar restrições e que afeta a eficácia do controle preditivo é a factibilidade do problema quadrático. Algumas vezes, durante a otimização, a região definida pelas restrições dos valores que as variáveis manipuladas podem assumir é um conjunto vazio. Nestas condições, o algoritmo de otimização não consegue encontrar nenhuma solução e o problema quadrático é dito infactível [5].

A infactibilidade da solução pode surgir tanto em regime permanente quanto durante os transitórios. Em regime permanente este problema surge, em geral, devido a objetivos de controles inatingíveis. Por exemplo, quando a referência não pode ser alcançada por causa das restrições nas variáveis manipuladas. Já nos transitórios, infactibilidade pode ocorrer caso haja uma perturbação ou uma mudança grande de referência que forçam o sistema de tal modo que não é possível trazê-lo de volta à região permitida com sinais de controle de amplitudes limitadas [5].

Como, havendo infactibilidade, o algoritmo MPC não consegue calcular as ações de controle, tem-se que tomar certas precauções. Em [5] são listadas várias técnicas para melhorar a factibilidade do problema quadrático como, por exemplo, desconectar temporariamente o controlador, mantendo um valor de controle pré-fixado, enquanto a factibilidade não é recuperada. Outras técnicas envolvem a remoção, ou apenas relaxação temporária das restrições violadas, recuperando assim a factibilidade.

Na versão atual do programa do sistema embarcado, utiliza-se a técnica de desconectar o controlador explicada anteriormente. Quando o programa detecta a condição de infactibilidade, mantém-se o último valor da ação de controle e, se esta condição se manter por um determinado período de tempo, altera-se o modo de controle para manual e um alarme é acionado. A técnica utilizada funciona bem quando a condição de infactibilidade for temporária, ou seja, o sistema naturalmente vai para uma região onde o problema quadrático se torna novamente factível. Como há várias situações onde isto não ocorre, futuramente será preciso adicionar técnicas que lidam com a infactibilidade de forma mais eficaz.

4.7 CONCLUSÕES

Este capítulo apresentou de forma detalhada como os algoritmos MPC implementados (GPC, DTCGPC e SSMPC) fazem uso do modelo do processo para calcular a ação de controle ótima a ser aplicada na planta. Também foi mostrado como obter a representação matricial das restrições mais comuns encontradas nos processos (limites das variáveis manipuladas, controlada e incrementos da ação de controle). Estas restrições podem ser aplicadas ao problema quadrático imposto pelos algoritmos MPC de forma a ter um controle maior do processo em malha fechada. No entanto, foi visto que o uso de restrições implica em um aumento do tempo de resolução do problema quadrático, além de possibilitar o surgimento de condições de infactibilidade que precisam ser tratadas adequadamente.

O próximo capítulo discutirá a metodologia de Engenharia de *Software* utilizada para implementar o programa do sistema embarcado e a interface de usuário.

5 PROJETO DE SOFTWARE

Este capítulo tratará da metodologia adotada para desenvolver o programa executado no sistema embarcado e a Interface de Usuário. A utilização de métodos da Engenharia de *Software* é essencial para a criação de programas eficientes, robustos e de fácil manutenção e extensão. Assim, a metodologia empregada no projeto será descrita em paralelo com os passos seguidos para a implementação tanto do programa do sistema embarcado quanto da interface.

5.1 PROJETO DE *SOFTWARE* DO SISTEMA EMBARCADO

Neste trabalho, optou-se por seguir a metodologia de análise e projeto de *softwares* descrita por Wazlawick em [50], que se baseia no Processo Unificado ou, em inglês, *Unified Process* (UP), apresentado em [51, 52]. O UP é bastante conciso e eficiente para a análise e projeto de sistemas orientados a objetos. No método, cada artefato utilizado para descrever o sistema (diagramas, tabelas, etc.) tem uma razão clara de existir, e as conexões entre os diferentes artefatos são muito precisas.

O Projeto Orientado a Objetos (POO) é um paradigma de programação onde o programa criado é modelado de modo similar àquele pelo qual as pessoas descrevem objetos reais. Ou seja, o uso da POO permite uma maneira natural e intuitiva de visualizar o processo de desenvolvimento de um *software*. Sistemas orientados a objetos são compostos de um número de objetos bem definidos e que se comunicam. Objetos com características e comportamentos em comum são organizados em classes [53]. Classes são as estruturas básica para modelar objetos e informações. Elas encapsulam atributos e métodos do objeto modelado que são, respectivamente, as características do objeto e as operações que o objeto pode realizar. O uso do POO produz programas mais inteligíveis, organizados e fáceis de manter, modificar e depurar [54]. Assim, o uso do POO em conjunto com uma boa metodologia de análise e projeto de *softwares* permite a criação de programas de forma eficaz e elegante cujas funcionalidades possam ser facilmente expandidas.

Um importante aspecto do UP é a forte associação à notação UML (*Unified Modeling Language*). A UML é uma linguagem gráfica que permite a representação padronizada de programas orientados a objetos [54]. Com seu uso, o entendimento e documentação do programa

são facilitados já que ela é mundialmente aceita.

O UP propõe um processo ágil, com poucos artefatos e pouca burocracia, o qual permite um desenvolvimento rápido. A documentação deve ser dirigida para a produção do *software*. Cada passo no processo tem um objetivo muito claro e uma utilização precisa, visando sempre à produção de um código que atenda aos requisitos do melhor jeito possível no menor tempo [50]. Este método é dividido em quatro fases: concepção, elaboração, construção e transição.

- Concepção: a primeira etapa de um projeto consiste na descrição detalhada dos requisitos do problema. Esta prática assegura que as características do sistema estejam bem definidas para as fases de projeto e desenvolvimento, diminuindo a chance de haver necessidade de mudanças no projeto durante a implementação, o que normalmente implica em custos e desperdício de tempo [54].

- Elaboração e Construção: a elaboração é constituída de análise e projeto, e a construção corresponde à implementação e aos testes. Estas fases são feitas de forma cíclica e iterativa, onde acontece a análise detalhada de uma parte do sistema (análise de requisitos e domínio) e nos quais é feito o projeto usando os padrões de projeto. Nos ciclos iterativos, são também feitos a implementação do código e os testes. Quando os ciclos iterativos terminam, tem-se então um sistema que está praticamente pronto, faltando apenas os testes finais de integração e de implantação junto ao usuário [50].

- Transição: ocorre após o último ciclo iterativo, quando o sistema, depois de pronto, será implantado.

A Fig. (24) mostra as fases da metodologia UP, identificando os ciclos iterativos.

5.1.1 Concepção

A fase de Concepção visa a compreensão abrangente, mas pouco profunda do sistema, de forma a levantar os requisitos básicos e fazer o planejamento do projeto. Deve-se buscar colher a maior quantidade de informações possíveis sobre as funções que o sistema deve executar e as restrições sob as quais deve operar. Estas restrições, também chamadas requisitos não-funcionais, especificam limitações a que o sistema está sujeito como, por exemplo no caso deste projeto, de que a ação de

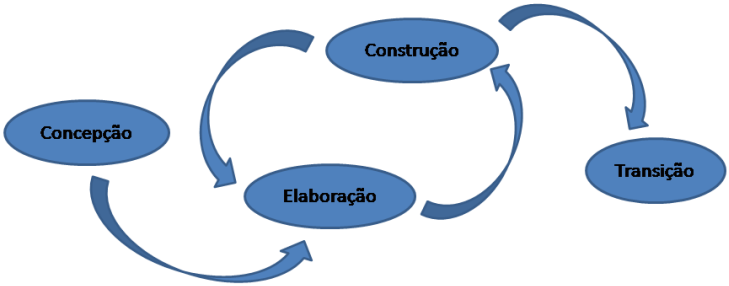


Figura 24 – Diagrama das fases do Processo Unificado.

controle a ser aplicada deve ser calculada dentro do período de amostragem, correndo o risco de, caso não obedecida a restrição, degenerar a resposta do sistema.

A fase de levantamento de requisitos deve ser uma fase de descoberta e não de invenção, deve-se listar o maior número possível de capacidades e restrições, mas sem se preocupar em ter uma lista completa. Requisitos não descobertos nesta fase deverão ser convenientemente acomodados ao longo do restante do processo de desenvolvimento [50]. Esta fase leva à produção de um Sumário Executivo, que é um texto corrido, simples, que fornece uma visão geral do sistema, e à listagem dos principais requisitos funcionais e não-funcionais. O Sumário Executivo deste projeto encontra-se na Tab. (3).

A partir do Sumário Executivo é feita uma listagem dos requisitos do sistema. Estes são numerados e organizados em tabelas como mostrado na Tab. (4), onde o requisito Calcular Ação de Controle é descrito. Em cada tabela há uma listagem dos requisitos não-funcionais associados e qualquer outra característica que possa ser pertinente. Foram encontradas funcionalidades que foram separadas em três categorias: (i) Algoritmo MPC; (ii) Comunicação com Processo e Usuário; (iii) Interação com Usuário.

As funcionalidades da categoria (i) são: (F1) Atualizar dados do processo, (F2) Calcular Ação de Controle, esta última mostrada na Tab. (4) e (F3) Alterar Modo de Controle. A categoria (ii) possui muitos requisitos não-funcionais que descrevem como a transmissão de dados é realizada e apenas um requisito nomeado (F4) Transmissão de Dados. Já a categoria de Interação com o Usuário descreve as seguintes funcionalidades: (F5) Configuração MPC, (F6) Configuração Rede Industrial, (F7) Verificação do Estado do Programa. As tabelas que descrevem cada um dos requisitos estão no Anexo A.

Tabela 3 – Sumário Executivo gerado na fase de Concepção do UP.

Sumário Executivo
Criar um *software* para o sistema embarcado capaz de executar algoritmos MPC e integrar com diversos tipos de protocolos de comunicação. O usuário deve ser capaz de configurar o tipo de algoritmo e seus parâmetros, e também que tipo de rede industrial usará para se comunicar com o restante do processo. O sistema deve ser capaz de detectar erros de comunicação com o processo e verificar de que forma estes prejudicarão o controle da planta, podendo até mesmo acionar alarmes através da rede. Ao ser executado, o programa deve obter informações pertinentes dos sensores e calcular a ação de controle a ser enviada ao processo, tudo através da rede industrial configurada. Como isto é feito por um algoritmo de alto custo computacional, o próprio sistema deve monitorar o tempo gasto para calcular a ação de controle e verificar se seu cálculo é factível dentro do tempo de amostragem especificado. Deverá existir a possibilidade de trocar o modo de controle para manual para interromper a operação de controle sem interromper o programa do sistema embarcado.

Tabela 4 – Tabela descrevendo o requisito funcional F2 - Calcular Ação de Controle.

F2 - Calcular Ação de Controle
Descrição - depois de o sistema obter as variáveis de processo necessárias, estas são utilizadas para o cálculo da ação de controle através do algoritmo MPC escolhido considerando as restrições dadas ao processo.
Restrições Não-Funcionais
NF1.1 - deve-se verificar o tempo necessário para o cálculo da ação de controle e, caso este extrapole o tempo de amostragem, deve-se tomar providências para que isto não ocorra.
NF1.2 - caso ocorra de o tempo de cálculo ultrapassar o tempo de amostragem seguidamente, um alarme deve ser acionado.
NF1.3 - caso ocorram problemas de otimização (ex. infactibilidade), deve-se tomar providências para não causar problemas de estabilidade.
NF1.4 - opção de vários algoritmos MPC.

Uma vez levantados os requisitos do sistema, é necessário agrupá-los em grupos correlacionados chamados Casos de Uso, que representam os principais negócios realizados pelo programa. Na Fig. (25) são mostrados os Casos de Uso encontrados: Configurar Controlador, Inicializar Controlador e Calcular Controle. Cada Caso de Uso é associado a um conjunto de requisitos funcionais do sistema e também são usados para indicar os atores envolvidos no conjunto de operações que é realizada, que, neste caso, são o Usuário e o Processo Industrial sendo controlado.

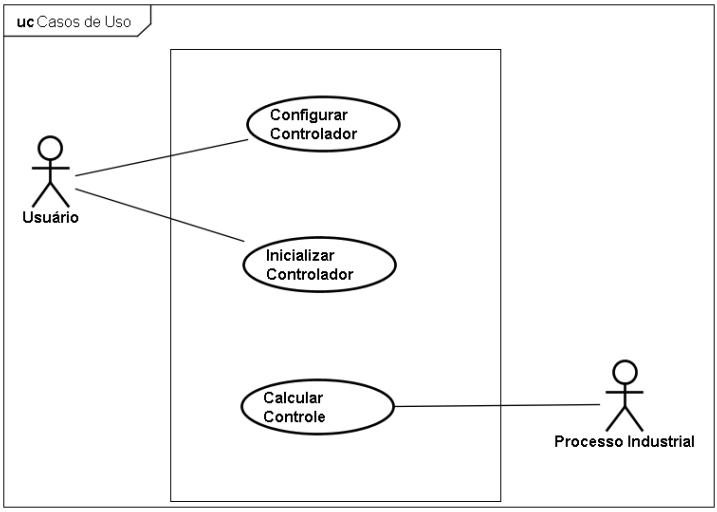


Figura 25 – Diagrama de Casos de Uso.

Com a obtenção dos Casos de Uso, é finalizada a fase de Concepção, onde se obteve de forma generalizada as funções que serão realizadas pelo *software* e os atores envolvidos em seu uso.

5.1.2 Elaboração e Construção

Cada ciclo iterativo dentro do Processo Unificado consiste em elaboração e construção. A fase de elaboração se inicia com uma subfase de análise e prossegue com a subfase de projeto. A fase de construção divide-se em implementação e teste do código produzido. A subfase de análise em si comporta três atividade distintas realizadas na seguinte

ordem: Expansão dos casos de uso, Construção do modelo conceitual, e Elaboração dos contratos das operações de sistema [50]. Já a subfase de projeto compõe-se das atividades de criação dos Diagramas de Classe e de Colaboração.

A primeira atividade da subfase de análise, a expansão dos casos de uso, corresponde ao aprofundamento da análise de requisitos. Já a modelagem conceitual corresponde à análise do programa em seus aspectos estáticos, ou seja, serão avaliados quais serão as informações complexas (conceitos) que serão representadas no sistema e depois darão origem às classes do programa. Os conceitos são ditos informações complexas porque não podem ser representados por simples tipos primitivos ou alfanuméricos como, por exemplo, números inteiros ou boleanos, ou uma simples *string*. Por fim, a elaboração dos contratos corresponde à especificação funcional dos aspectos dinâmicos, ou interação entre conceitos, do programa. Os artefatos gerados na realização dos passos citados da fase de análise serão utilizados na fase de projeto.

5.1.2.1 Expansão dos Casos de Uso

Quando se está expandindo um caso de uso é preciso proceder a um exame detalhado do processo sendo realizado. Deve-se descrever o caso de uso passo a passo: como ele ocorre, como é a interação entre os usuários e o sistema. É preciso identificar o Fluxo Principal do caso de uso, na qual se descreve o que acontece quando tudo dá certo na interação, as variantes do fluxo principal, que são fluxos alternativos tomados devido à característica da informação com que se está lidando, e, por fim, as exceções, que definem o que ocorre caso haja algum entrave que não permita o seguimento normal no fluxo principal ou variante. Na Tab. (5) é mostrado a expansão do caso de uso Calcular Controle.

5.1.2.2 Modelagem Conceitual

O modelo conceitual deve descrever e compreender a informação que o sistema vai gerenciar, assim, representa somente o aspecto estático da informação, não havendo referências a operações ou outros aspectos dinâmicos do sistema projetado [50]. Os conceitos representados no modelo conceitual são representações da informação complexa, que não pode ser descrita meramente por tipos alfanuméricos. Os conceitos

Tabela 5 – Tabela descrevendo a expansão do caso de uso Calcular Controle.

Caso de Uso: Calcular Controle

Fluxo Principal	**Fluxos Alternativos**
1. A cada período de amostragem do processo o sistema consulta as variáveis de saída (saídas, perturbações e referências) através da rede industrial. 2. O sistema gera a ação de controle. 3. O sistema envia a ação de controle para o processo através da rede industrial.	2.a Modo Manual: utiliza-se as ações de controle manuais enviadas pelo usuário. 2.b Modo Automático: o sistema utiliza os dados do processo para gerar as matrizes de restrições, caso houver, e da resposta livre de acordo com as particularidades do algoritmo MPC sendo utilizado. **Exceções** 1.a Alguns dos dados estão desatualizados: deve-se mudar o modo de operação para Manual e ativar alarme.

geralmente possuem vários atributos, estes sim alfanuméricos, que o caracterizam. Há também as associações, que representam um tipo de informação que liga diferentes conceitos entre si.

Existem diferentes métodos para a descoberta dos conceitos, mas a forma que se utilizou foi a análise dos textos dos casos de uso expandidos. Nesta análise, procurou-se descobrir todos os elementos textuais que referenciam informações relevantes para o sistema que devem ser guardadas [50].

Na Fig. (26) é mostrado o modelo conceitual utilizado no projeto. Foram encontrados os seguintes conceitos:

- Sistema: não é um conceito em si, é utilizado apenas para representar o sistema com um todo;
- Modelo: representa as informações da planta a ser controlada.

Atributos: ponto de operação do modelo, tipo de representação(ex. função de transferência) e seus parâmetros;

- Rede Industrial: representa a rede sendo utilizada pelo programa para se comunicar com o processo real e armazena os dados recebidos/enviados. Atributos: nome ou identificador, tipo de rede, parâmetros de configuração;

- Processo: representa o processo sendo controlado, a ele está associado o controlador MPC, os instrumentos da planta e os modelos utilizados. Atributos: nome, período de amostragem, modo de controle (Manual/Automático), quantidade de entradas, saídas e perturbações, estado do processo, tipo de representação (linear, linearizado ou não-linear);

- Instrumento: representa os instrumentos presentes no processo. Este conceito encapsula o processo de conversão e normalização de dados recebidos/transmitidos pela rede. Atributos: nome ou identificador, tipo de instrumento (sensor, atuador, alarme), unidade, valor indicado pelo instrumento, parâmetros para conversão dos dados da rede;

- Interface Rede Industrial: é a interface pela qual o conceito Instrumento recebe/envia dados pela rede. Com a separação entre interface de rede e instrumento, é possível alterar o tipo de rede de um instrumento modificando apenas a sua interface associada. Há também a vantagem de que qualquer alteração na implementação do acesso à rede não resulte numa alteração da classe Instrumento. Atributos: tipo de rede, parâmetros de configuração do acesso aos dados recebidos/enviados pela rede;

- Controlador: representa o algoritmo MPC sendo utilizado, está associado ao modelo, pois é a partir deste que o controlador é calculado. Atributos: horizontes de controle e predição, ponderações, restrições do processo.

Ao final da construção do modelo conceitual, obtem-se uma representação fiel e organizada da informação gerenciada pelo *software* a ser criado.

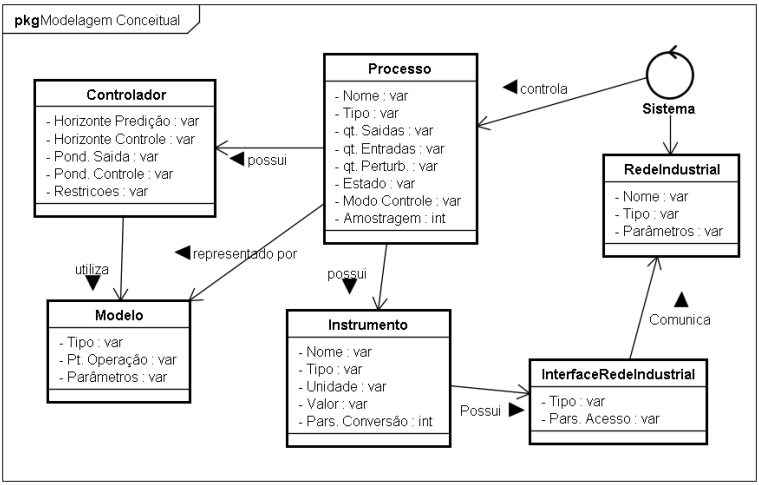


Figura 26 – Modelo Conceitual do projeto.

5.1.2.3 Contratos

A partir dos casos de uso é possível identificar as operações e consultas de sistema. As operações de sistema correspondem a inserções de informação no programa e as consultas de sistema correspondem ao acesso à informação armazenada. Cada operação ou consulta deste tipo implica a existência de uma intenção por parte dos atores, a qual é capturada pelos contratos de operações e consultas de sistema. Os contratos são basicamente tabelas onde se identificam precondições, pós-condições e exceções no caso de um contrato de operação e precondições e resultados no caso de um contrato de consulta. As precondições definem o que deve ser verdadeiro na estrutura da informação armazenada para a operação ou consulta a ser executada. As pós-condições definem o que muda na informação armazenada após a execução da operação. Resultados são os dados obtidos por uma consulta ao sistema, ou seja, não se pode alterar dados, pois isto está restrito a operações. Por fim, as exceções, que se referem a condições que devem ser avaliadas durante a execução da operação e que, dependendo da avaliação, a operação não poderá ser concluída [50].

Um artefato utilizado em conjunto com os contratos são os diagramas de sequência, que podem ser construídos a partir dos casos de uso. Estes diagramas têm como elementos instâncias de atores, repre-

Tabela 6 – Tabela que mostra o contrato de uma das funções da Interface de Usuário.

Interface de Usuário
operação: DefineAlgoritmoMPC(tipoMPC:String)
precondições: deve existir uma instância de processo
pós-condições: foi criado uma nova instância da classe MPC definida pelo variável tipoMPC. A instância criada foi associada à instância de Processo corrente.

sentados por figuras humanas esquematizadas, e instâncias de objetos constituintes do sistema. Cada um destes elementos terá uma linha de tempo, representada pelas linhas verticais, na qual os eventos podem ocorrer. Quando a linha está tracejada, indica inatividade por parte do ator ou sistema. As linhas horizontais representam fluxo de informação.

Na Fig. (27) é mostrado o diagrama de sequência para se configurar um novo processo no sistema embarcado. Neste diagrama são definidas a ordem de execução das consultas e operações, e também se há repetições de uma mesma função, como, por exemplo, o item 8 que é repetido enquanto houver instrumentos a serem definidos. Na Tab. (6) é mostrada o contrato da operação 3 do diagrama da Fig. (27).

5.1.2.4 Diagramas de Colaboração e de Classes

A fase de projeto do *software* visa produzir uma solução para o problema agora claramente identificado pela análise. O problema está especificado no modelo conceitual, nos contratos e nos diagramas de sequência dos casos de uso. No momento, resta projetar uma arquitetura de *software* para realizar concretamente o sistema [50].

A atividade de projeto pode ser dividida em tarefas com diferentes objetivos. Em primeiro lugar, é importante realizar o projeto da camada de domínio do *software*. Esta camada corresponde ao conjunto de classes que vai realizar toda a lógica do sistema de informação; as demais camadas (persistência, interface, segurança etc.) são derivadas ou dependentes da camada de domínio e serão vistas nas seções seguintes.

O projeto da camada de domínio compõe-se basicamente de duas atividades: (i) definir os diagramas de colaboração e (ii) elaborar o diagrama de classes. Os diagramas de colaboração se baseiam nos diagramas de sequência e indicam detalhadamente como é feita a comunicação, através de mensagens, entre instâncias de objetos das diferen-

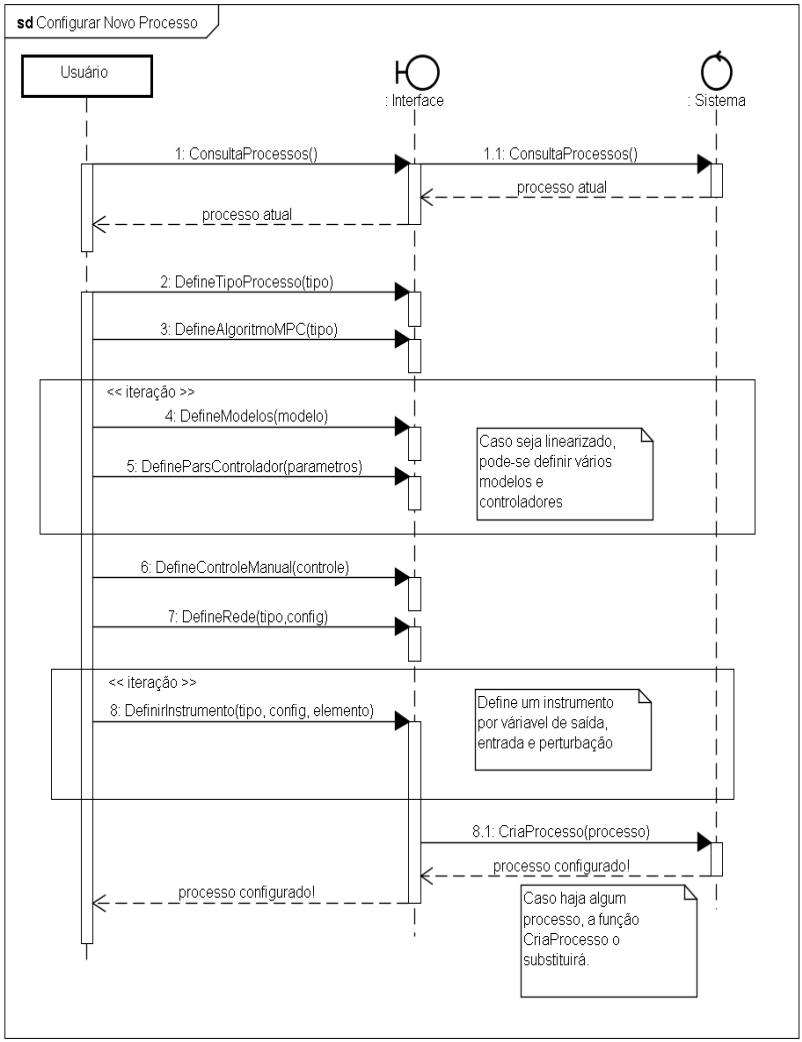


Figura 27 – Diagrama de sequência para a configuração de um novo processo.

tes classes projetadas. Estas mensagens definirão os métodos que serão implementados nas classes do projeto e podem ser básicas ou delegadas. As primeiras ocorrem quando o objeto que recebe a mensagem

é responsável por executar a operação/consulta requisitada. Já as delegadas ocorrem quando a mensagem recebida é passada adiante para um outro objeto que realizará esta tarefa. Com o uso de delegação de tarefas, consegue-se o que se chama padrão acoplamento fraco [55], ou seja, evita-se ao máximo a criação de novas associações entre objetos no diagrama de colaboração, pois isso pode causar o aumento de complexidade das classes envolvidas. Por exemplo, caso seja necessária a alteração de uma classe, geralmente é necessário também fazer alguma modificação nas classes associadas, assim, quanto menor o número de associações, menor a quantidade de modificações o projeto necessitará.

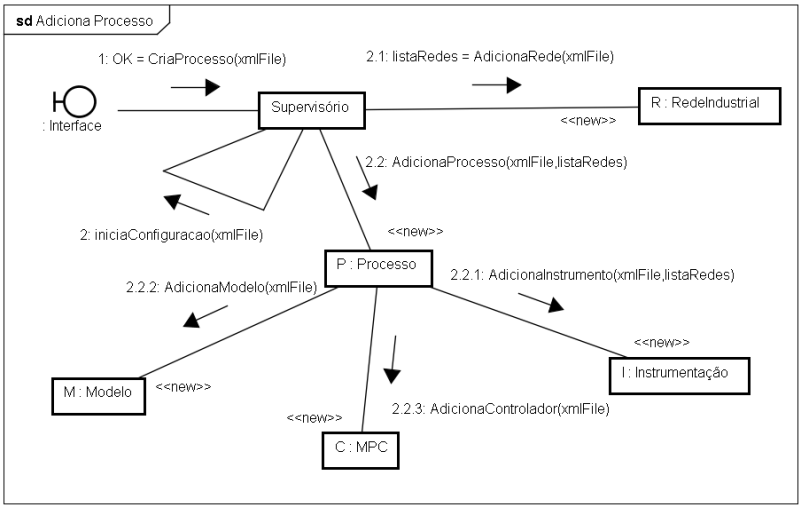


Figura 28 – Diagrama de colaboração entre as entidades do programa quando o usuário requisita a criação de um novo processo.

Na Fig. (28) é mostrado o diagrama de colaboração entre as instâncias das classes ao se criar um novo processo. Inicialmente a interface faz a requisição de criação através da chamada função CriaProcesso, que têm como parâmetro o arquivo XML de configuração do processo. Recebida esta mensagem, a instância de Supervisório faz uma chamada da função IniciaConfiguracao que irá dar continuidade ao procedimento de criação do novo processo. Cria-se, primeiramente, as redes industriais, que são armazenadas na variável listaRedes, depois se instancia um objeto da classe Processo, que recebe como parâmetro o arquivo XML e a lista de redes. Para finalizar, o objeto de Processo

cria as instâncias de Modelo, MPC e Instrumento necessárias para o seu funcionamento.

Depois de concluídos os diagramas de colaboração é possível criar o diagrama de classes, pois detalhes como métodos a serem inseridos e direção das associações existentes só podem ser efetivamente inseridos no diagrama após a fase de elaboração dos diagramas de colaboração [50]. Num primeiro momento, o diagrama de classes é uma cópia exata do modelo conceitual, depois há a adição de métodos e da direção das associações, obtidos através dos diagramas de colaboração. Também haverá detalhamento maior dos atributos e, se for necessário, alterações na estrutura das classes e associações. Geralmente também são criados atributos privados, ou seja, informações que representam a dinâmica interna do próprio objeto, assim, não faz sentido expô-los a outros objetos.

Na Fig. (29) é mostrado o diagrama de classes da última versão do *software* do sistema embarcado. Foram retirados os atributos e métodos de forma que fosse possível visualizar a topologia completa do sistema. Os métodos e atributos são detalhados no Anexo A. Nota-se a adição de várias classes que não existiam no modelo conceitual e do uso de associações por herança, que são identificadas pelas setas cheias no diagrama de classes. Herança é obtida quando duas classes se associam por meio de uma relação especial denominada generalização (no sentido da classe mais específica – subclasse; para a mais genérica – superclasse). A generalização deve ser usada sempre que um conjunto de classes $(X_1, \ldots, X_n)$ possuir diferenças e semelhanças específicas, de forma a possibilitar o agrupamento das semelhanças em uma superclassse X, e as diferenças nas subclasses $X_1, \ldots, X_n$. Um exemplo é mostrado na Fig. (30), onde a superclasse *Instrumento* possui atributos e métodos comuns a todos os tipos de equipamentos, tais como o nome e a unidade de engenharia utilizada, mas ela é subdividida em três outras classes: Sensor, Atuador e Digital. Esta subdivisão ocorre porque há características distintas em cada uma destas classes como, por exemplo, no caso do Sensor. Um objeto Sensor lê os dados provenientes do processo e, se necessário, filtra os dados obtidos e só depois disponibiliza estes para o controlador. Esta característica não tem motivo de existir num equipamento do tipo Digital ou Atuador.

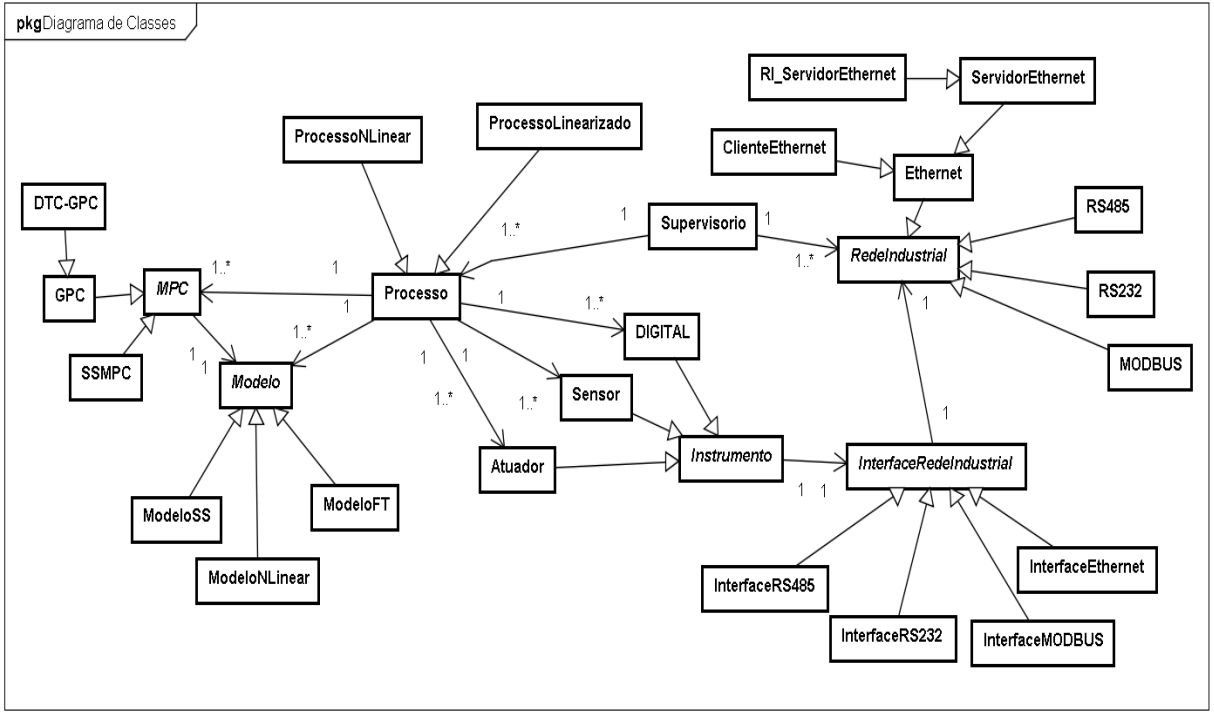


Figura 29 – Diagrama de classes do *software* criado para o sistema embarcado.

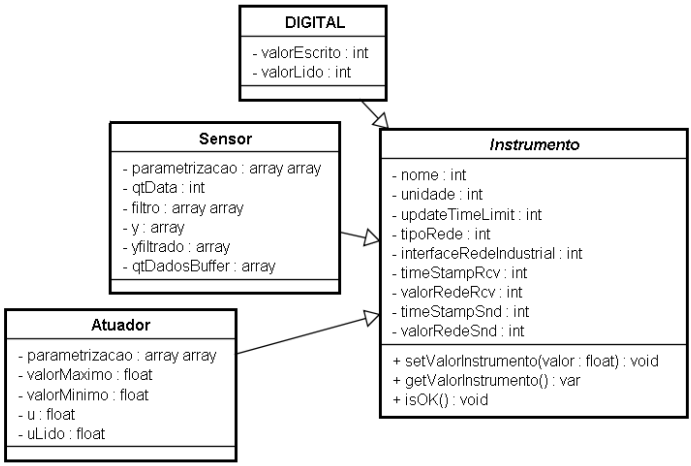


Figura 30 – Exemplo de associação por herança.

A seguir, uma descrição do elementos e associações presentes no diagrama de classes da Fig. (29):

- Supervisório: classe responsável pela comunicação com a interface de usuário, gerenciamento das redes industriais e processos. A comunicação com o usuário é feita exclusivamente por rede TCP/IP Ethernet através de um servidor local implementado pela respectiva classe;

- Rede Industrial: superclasse que implementa redes industriais. Tem como subclasses: Ethernet, RS485, RS232 e MODBUS. A classe Ethernet também possui classes derivadas que implementam características diferentes da comunicação por este tipo de rede;

- Processo: classe que faz o gerenciamento do controle de um processo. Cada instância de Processo está associada a objetos Sensor, Atuador e Digital, através dos quais realiza a comunicação com o processo real. Há uma associação também com um controlador preditivo, que é um objeto da classe *MPC*, e com um modelo da planta (classe *Modelo*). Caso o processo real possua um modelo linearizado, a classe derivada ProcessoLinearizado deve ser usada, pois esta realiza a linearização das variáveis, e, no caso de um modelo não-linear, usa-se a classe ProcessoNLinear. No caso linearizado, pode haver a associação a vários controladores MPC

que são automaticamente trocados quando há a troca de ponto de operação;

- Modelo: superclasse abstrata que representa o modelo do processo. Os modelos podem ser não-lineares, por função de transferência e por espaço de estados, que são representados, respectivamente, pelas classes ModeloNLinear, ModeloFT e ModeloSS;

- MPC: superclasse abstrata que representa um controlador preditivo genérico. Suas classes derivadas são SSMPC, GPC e DTCGPC que implementam os respectivos algoritmos MPC. Cada controlador está associado a um único modelo da planta;

- Instrumento: superclasse abstrata que converte e processa os dados provenientes da rede industrial. Suas classes derivadas são: Sensor, Atuador e Digital. Esta última é usada para implementar alarmes e a troca do modo de controle (Manual/Automático). Cada instância desta classe está associada a um objeto InterfaceRedeIndustrial que obtem o dado sem processamento da rede industrial;

- InterfaceRedeIndustrial: cada interface está associada a uma instância de RedeIndustrial específica. Dependendo do tipo de rede, usa-se uma das classes derivadas (InterfaceRS485, InterfaceRS232, InterfaceMODBUS, InterfaceEthernet) que implementam o acesso à rede industrial.

5.1.2.5 Camada de Persistência

A camada de persistência é a parte do programa responsável por automatizar o salvamento e a recuperação de dados do *software* em uma mídia não-volátil, tal como um HD. Desta forma os dados podem ser recuperados mesmo que o sistema seja desligado ou reiniciado.

Para este projeto, inicialmente especulou-se a necessidade de armazenagem de dois tipos de dados: os de configuração do processo (controlador, instrumentos, redes, etc.) e histórico das variáveis de interesse da planta. O primeiro tipo é essencial e foi mantido. Quanto ao segundo, verificou-se que em sistemas distribuídos de controle sempre há um sistema secundário responsável pela captação e armazenagem do histórico das variáveis de interesse e, como o controlador MPC precisa de um número relativamente pequeno de dados para funcionar, optou-se por não manter em memória não-volátil este tipo de dado.

A armazenagem dos dados de interesse é feita no formato XML (*Extensible Markup Language*), que é uma linguagem de marcação que define um conjunto de regras para a geração de documentos de tal forma que estes sejam legíveis tanto por humanos quanto para máquinas [56]. Ela é mantida pela W3C, a principal organização de padronização da *World Wide Web*, e todas as especificações são abertas e livres de licenças.

Documentos XML geralmente são interpretados utilizando pacotes de *software* específicos chamados XML *parsers* (neste projeto utilizou-se a biblioteca Python MiniDom [57]). O XML utiliza *tags* ou marcações para determinar o tipo de objeto sendo armazenado, assim como o HTML, com a diferença de que é possível criar marcações específicas para cada aplicação. Assim, pode-se criar elementos de qualquer tipo e combiná-los em elementos diferentes. A estrutura hierárquica do XML torna possível a aplicação do conceito de programação orientada a objetos aos documentos, assim, é extremamente simples criar uma camada de persistência utilizando XML, bastando apenas definir os tipos dos elementos utilizados [58]. Por exemplo, na Fig. (31) é mostrado um trecho de documento XML onde define-se um elemento Processo que possui os seguintes elementos como atributos: Tipo, Nome, Amostragem e Modo. O valor que cada um destes atributos assume está entre as marcações correspondentes (texto em preto).

```
<?xml version="1.0" encoding="UTF-8"?>
<Processo>
    <Tipo>Linear</Tipo>
    <Nome>HysysColuna</Nome>
    <Amostragem>60</Amostragem>
    <Modo>AUTO</Modo>
</Processo>
```

Figura 31 – Exemplo de um Processo sendo definido em um arquivo XML.

Outra característica importante do XML é a validação de documentos. É possível especificar *a priori* como os documentos XML utilizados devem ser construídos e interpretados. Assim, os documentos só são realmente utilizados se estiverem dentro de um formato padrão, evitando assim erros devido a arquivos corrompidos ou incompletos, e exceções dentro do programa devido a dados incorretos. Este formato

é definido em um documento onde se define os tipos de elementos, se estes são básicos (texto, número inteiro ou fracionário, etc.) ou se são complexos (compostos de outros elementos), a quantidade de cada elemento e a posição em que devem aparecer, ou seja, é possível definir completamente a estrutura que os documentos devem seguir.

Os documentos XML que definem estas características são chamados XML *schemas*. Existem diferentes linguagens para a definição dos *schemas* sendo que as mais comuns são o DTD (*Document Type Definition*) e o *XML Schema*. Este último foi escolhido para ser utilizado no projeto pois permite grande precisão e flexibilidade na descrição da estrutura e dos elementos do documento [58]. Assim como o próprio XML, o *XML Schema* é mantido pela W3C.

Com o uso de arquivos XML, a configuração do sistema embarcado fica simples e fácil de implementar, pois basta o envio do arquivo de configuração, que define todas as características do processo, de comunicação e controle, para o programa do sistema embarcado pela Interface de Usuário.

O *schema* utilizado neste projeto foi criado através do uso do programa Altova XMLSpy, que fornece uma interface de usuário que facilita a geração de *schemas* e de documentos XML. O *schema* que define o documento de configuração do sistema embarcado pode ser visualizado em [59].

5.1.2.6 Implementação

Com os documentos gerados até este ponto no projeto, a geração de código se torna quase automática. A partir deles são criados os códigos das classes correspondentes à camada de domínio da aplicação, ou seja, as classes que realizam toda a lógica do sistema a partir das operações e consultas de sistema. Também é feito a integração entre a camada de persistência e domínio, para que os dados de configuração do sistema possam ser armazenados e recuperados quando necessário.

Assim, criou-se um *software* para o sistema embarcado cujo funcionamento será descrito a seguir. Ao ser iniciado, o programa cria um objeto da classe Supervisório (ver Fig. (29)). Esta classe, como descrito anteriormente, é responsável pela comunicação com a interface de usuário e com o processo, pela criação do objeto da classe Processo, que fará o controle da planta, e também cria objetos responsáveis pelas redes industriais. Os objetos das classes Supervisório, Processo e RedeIndustrial implementam tarefas ou *threads* que, no contexto de

um programa, são trechos de *software* que são executados de forma paralela, o que não significa que elas operam independentemente. Por exemplo, o controle feito pelo objeto Processo só terá validade se a rede industrial estiver funcionando.

Depois de iniciado, o objeto de Supervisório cria um servidor Ethernet que espera mensagens da interface de usuário. Esta mensagem pode solicitar que um novo processo seja criado, que um processo existente entre em funcionamento, e que um processo seja interrompido. Os efeitos das diferentes mensagens são mostrados no diagrama de estados da Fig. (32), onde se mostra de forma simplificada os passos executados pela tarefa do Supervisório.

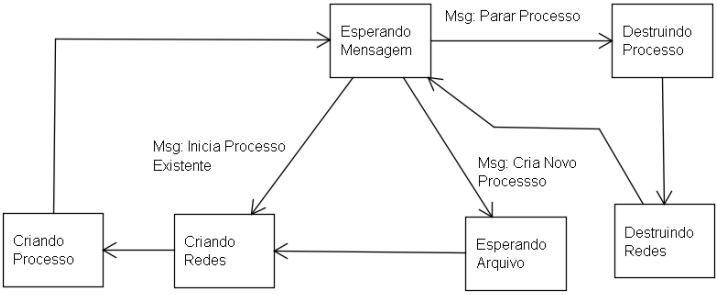


Figura 32 – Diagrama de estados da classe Supervisório.

Já na Fig (33) é mostrado o diagrama de estados de um objeto da classe Processo. Depois de iniciado, são criados os objetos *Instrumento* da planta que enviarão/receberão dados do processo, e também os objetos das classes *Modelo* e *MPC*. Após estas operações, os dados das variáveis de processo e de modo de controle são atualizadas. Em seguida, dependendo do modo de controle, ocorre operações diferentes. No caso de modo Manual, o valor de controle inserido pelo usuário é obtido e depois repassado aos atuadores. Caso o modo seja Automático, calcula-se o valor das variáveis manipuladas através do algoritmo MPC configurado e depois atualiza-se o valor dos atuadores. Por último, caso ocorra uma situação atípica em que seja acionado o alarme, o processo passa automaticamente para modo manual. Em todos os casos, após um período de amostragem, volta-se ao estado onde se atualiza as variáveis de processo, reiniciando o ciclo.

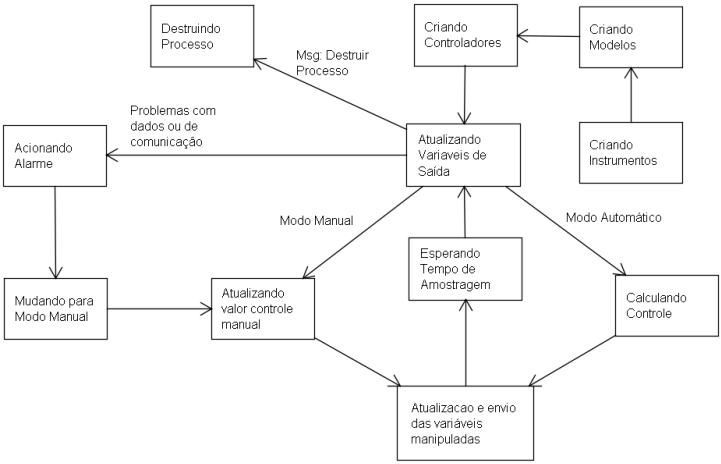


Figura 33 – Diagrama de estados da classe Processo.

5.2 INTERFACE DE USUÁRIO

A versão atual da Interface de Usuário é bastante simples e é mostrada na Fig. (34). Este programa foi criado utilizando a linguagem Java através do *software* Netbeans, que fornece um ambiente de desenvolvimento que visa facilitar a criação de aplicativos com interface gráfica Java.

Na Fig. (34) é mostrada a tela de adição de novos processos, onde se especifica a quantidade e os nomes das entradas, saídas e perturbações, também se configura o nome do processo, o tipo e o período de amostragem a ser utilizado. Já na Fig. (35) está a janela de adição e configuração de redes, onde se pode adicionar quantas redes industriais forem necessárias, inclusive de tipos diferentes. Após a configuração das redes, segue-se a tela de interfaces de rede, onde se associa cada variável de saída, atuação e perturbação a uma rede industrial, depois são configurados os modelos e os controladores. Para finalizar, é gerado o arquivo XML a ser enviado para o sistema embarcado para implementar o controle do processo.

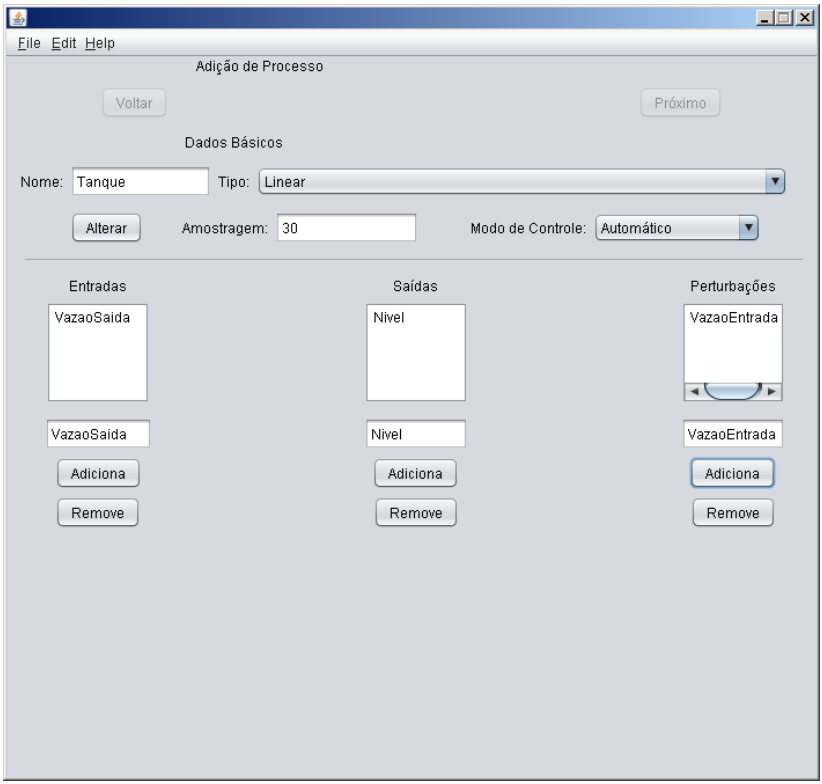


Figura 34 – Tela de adição de um novo processo da Interface de Usuário.

5.3 CONCLUSÕES

Neste capítulo foi mostrada a metodologia adotada para se desenvolver o *software* do sistema embarcado e da interface de usuário. A metodologia adotada, o Processo Unificado, é pragmática, pois permite a criação de programas de forma ágil a partir de artefatos UML que têm utilidade clara na implementação do projeto, evitando, assim, geração de documentação desnecessária e aumento do tempo de projeto. Com esta metodologia foi possível criar um programa confiável, de fácil manutenção e extensão.

Quanto à interface de usuário, suas funcionalidades ainda estão limitadas à criação e configuração de um processo e sua inicialização no sistema embarcado, mas, em versões posteriores, pretende-se criar

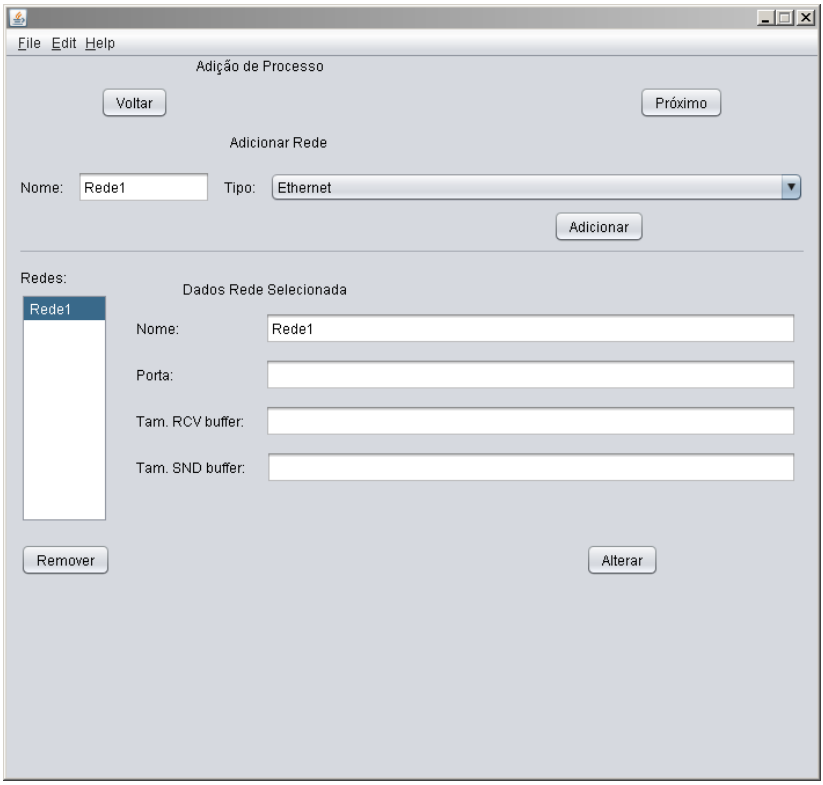


Figura 35 – Tela de adição e configuração de redes industriais.

uma interface com mais funções que agilizem o processo de ajuste do controlador, como, por exemplo, a adição de um ambiente de simulação do modelo da planta para verificar se o ajuste feito atende as especificações dinâmicas e estáticas da planta. No próximo capítulo serão vistos os resultados experimentais envolvendo os programas criados.

6 RESULTADOS EXPERIMENTAIS

Este capítulo apresentará alguns experimentos concebidos de modo a validar o *software* criado para o sistema embarcado e a aplicabilidade do protótipo em plantas com dinâmicas lentas de médio e pequeno porte.

6.1 AVALIAÇÃO DO PROTÓTIPO

Os dois primeiros experimentos foram realizados para verificar a capacidade computacional do protótipo, e também para demonstrar como seria feita a avaliação da possibilidade de se utilizar o sistema embarcado desenvolvido em determinado processo. Para fazer esta avaliação, fez-se experimentos *Hardware-in-the-Loop* onde se variou os horizontes de predição e controle do controlador, e a quantidade de restrições ativas, de forma a verificar como estas características alteram o tempo de cálculo da ação de controle. Vale ressaltar que o tipo de algoritmo MPC utilizado não influencia o tempo necessário para resolver o problema quadrático, pois não é o tipo de algoritmo que define o tamanho do problema a ser resolvido, e sim os horizontes e restrições utilizadas.

Depois de descrever o que é um experimento *Hardware-in-the-Loop*, serão mostrados os exemplos usados, que foram retirados de livros sobre controle preditivo, e os parâmetros de simulação. Por último, será feita uma análise dos resultados.

6.1.1 Experimento *Hardware-in-the-Loop*

Para realizar os experimentos da forma mais realista possível, utilizou-se a técnica *Hardware-in-the-Loop* (HIL), que tem sido usada na indústria de Defesa e Aeroespacial desde 1950, mesmo tendo um custo elevadíssimo para a época. Isto porque havia um sério risco de vidas humanas ao se testar os equipamentos sendo desenvolvidos nos processos reais. Com o avanço da tecnologia e da facilidade de se conseguir equipamentos eletrônicos de baixo custo a pronta-entrega, a utilização da simulação HIL ganhou muito espaço, principalmente na indústria automobilística na década de 1990 [60].

A ideia básica do HIL é simples. Dentro do ambiente HIL, o

equipamento a ser testado, que neste caso é o sistema embarcado MPC, interage com um sistema virtual que substitui o sistema real. A maior vantagem do uso da simulação HIL é que o equipamento pode ser testado em uma variedade de cenários de operação sem que seja necessário: (i) o uso do sistema real ou a construção de um modelo em escala do sistema; (ii) a criação de um modelo virtual do equipamento a ser testado para que haja validação [61]. Além disso, também pode ser utilizado em treinamento de operadores de campo.

O experimento HIL é geralmente utilizado quando é possível modelar a dinâmica do processo com grande precisão e fidelidade, e quando experimentos com o processo real implicariam em custos desnecessários devido ao tempo de instalação e testes do novo equipamento. Além disto, em muitos casos, é uma forma de evitar riscos desnecessários em projetos onde a segurança das pessoas pode ser prejudicada [62].

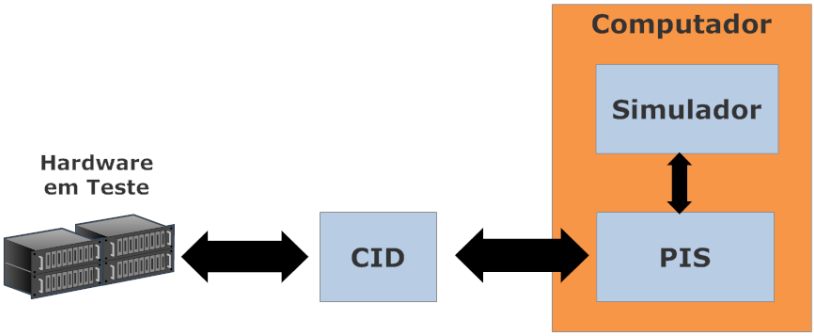


Figura 36 – Esquema da simulação HIL.

Assim como é mostrado na Fig. (36), pode-se dividir o HIL em três componentes [63]:

1. Dispositivo Controlador de Interface ou, em inglês, *Controller Interface Device* (CID);

2. Um módulo de *software* chamado Programa de Interface da Simulação (PIS) que fará a ligação entre o CID e o programa de simulação;

3. Um Simulador que é responsável por gerar as saídas do processo de acordo com as entradas fornecidas pelo controlador sendo testado e que é executado em um computador hospedeiro .

O primeiro componente, o CID, é a interface pela qual o equipamento sendo testado envia e recebe todas as informações necessárias para realizar a tarefa para a qual foi desenvolvido. Geralmente o CID é um equipamento eletrônico que faz a conversão de dados analógicos para um formato digital transmissível por um tipo rede (Ethernet, RS232, etc.) para o computador hospedeiro e vice-versa. No computador hospedeiro estão os componentes restantes, o PIS e o Simulador. A função deste último é auto-explicativa, já o PIS tem como objetivo coordenar como a simulação é feita no Simulador e também realizar a troca de dados com o CID.

No caso dos experimentos iniciais, o equipamento em teste, o protótipo, faz a transmissão somente de dados digitais através de redes industriais, assim, o CID não é necessário. A rede industrial utilizada nesses experimentos foi a Ethernet com um protocolo baseado em TCP/IP feito pelo autor para facilitar os testes com o sistema embarcado. Apesar de não ser uma rede industrial *de facto*, seu uso não invalida os resultados pois estes são independentes do tipo de rede em uso.

Quanto ao Simulador e ao PIS, estes foram implementados na forma de programas Python. O Simulador foi feito de tal forma que é possível adicionar ruído ao processo simulado, variar os parâmetros do modelo da planta para simular erros de modelagem e inserir perturbações nas entradas ou saídas do processo. O PIS, como explicado, faz o gerenciamento dos dados trocados entre planta e protótipo.

6.1.2 Experimento 1: Fracionador de Óleo Pesado

O modelo Fracionador de Óleo Pesado da Shell, apresentado em [64], tem sido utilizado na literatura para o teste de diferentes estratégias de controle para colunas de destilação e em [5, 41], foi utilizado para analisar algoritmos MPC. Este exemplo ilustra o controle de processos MIMO com diferentes atrasos nas variáveis de saída.

O processo, mostrado na Fig. (37), possui três variáveis controladas: as composições dos produtos de topo (y_1) e intermediário (y_2), que são medidos por analisadores, e a temperatura de fundo (y_3). As variáveis manipuladas são as vazões de topo (u_1) e lateral (u_2) e a porcentagem de refluxo no fundo da coluna (u_3). O modelo contínuo do processo é representado pelas seguintes funções de transferência, onde os tempos são dados em minutos:

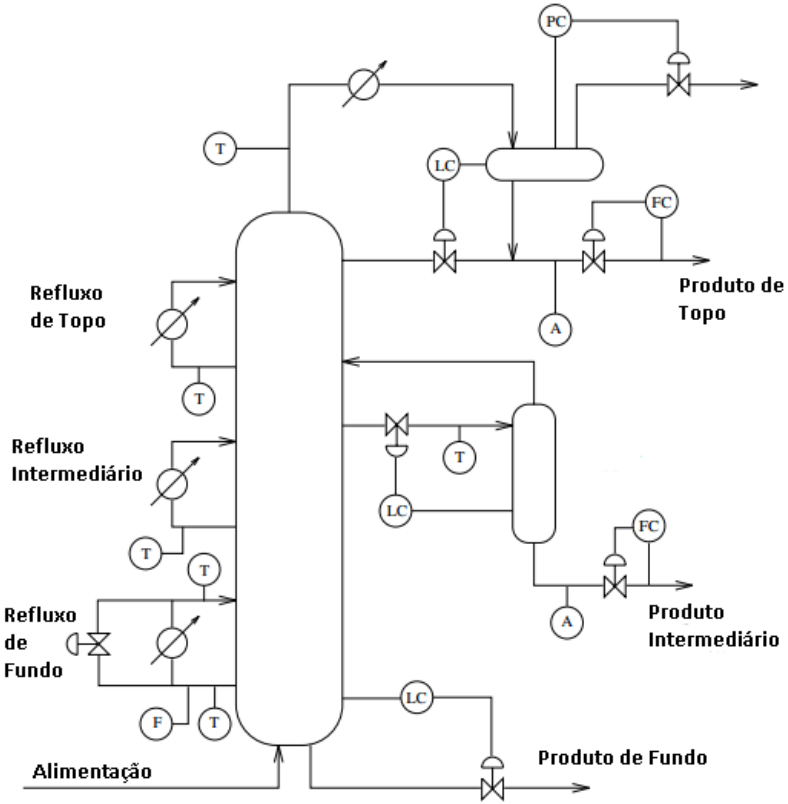


Figura 37 – Esquema do Fracionador de Óleo Pesado do exemplo 1.

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \\ Y_3(s) \end{bmatrix} = \begin{bmatrix} \frac{4{,}05e^{-27s}}{1+50s} & \frac{1{,}77e^{-28s}}{1+60s} & \frac{5{,}88e^{-27s}}{1+50s} \\ \frac{5{,}39e^{-18s}}{1+50s} & \frac{5{,}72e^{-14s}}{1+40s} & \frac{6{,}9e^{-15s}}{1+40s} \\ \frac{4{,}38e^{-20s}}{1+33s} & \frac{4{,}42e^{-22s}}{1+44s} & \frac{7{,}2}{1+19s} \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \\ U_3(s) \end{bmatrix}. \quad (6.1)$$

Discretizando as funções de transferência da equação anterior com um período de amostragem de $T_s = 4$ min, tem-se o modelo discreto do sistema:

$$\begin{bmatrix} \frac{0{,}08(z^{-1}+2{,}88z^{-2})}{1-0{,}923z^{-1}}z^{-6} & \frac{0{,}114z^{-1}}{1-0{,}936z^{-1}}z^{-7} & \frac{0{,}116(z^{-1}+2{,}883z^{-2})}{1-0{,}923z^{-1}}z^{-6} \\ \frac{0{,}211(z^{-1}+0{,}96z^{-2})}{1-0{,}923z^{-1}}z^{-4} & \frac{0{,}187(z^{-1}+0{,}967z^{-2})}{1-0{,}936z^{-1}}z^{-3} & \frac{0{,}17(z^{-1}+2{,}854z^{-2})}{1-0{,}905z^{-1}}z^{-3} \\ \frac{0{,}5z^{-1}}{1-0{,}886z^{-1}}z^{-5} & \frac{0{,}196z^{-1}+0{,}955z^{-2}}{1-0{,}913z^{-1}}z^{-5} & \frac{1{,}367z^{-1}}{1-0{,}81z^{-1}} \end{bmatrix}. \tag{6.2}$$

Para este processo, três testes foram feitos onde se adotou o algoritmo DTC-GPC. Em todos eles os pesos são constantes: $\boldsymbol{Q_y} = diag(\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{I})$ e $\boldsymbol{Q_u} = diag(0,3\boldsymbol{I}; 0,3\boldsymbol{I}; 0,5\boldsymbol{I})$. Foram utilizadas restrições no valor absoluto da ação de controle e no seu incremento: (i) as entradas podem variar no intervalo [-0,5;0,5] e (ii) incremento máximo de $\pm 0,05/min$ ou $\pm 0,2$ por período de amostragem. As restrições utilizadas e os horizontes sofrem variações:

- Teste 1: $N_i = 30$, $\forall i$, $N_{ui} = 15$, $\forall i$, e restrições (i) e (ii);
- Teste 2: $N_i = 30$, $\forall i$, $N_{ui} = 15$, $\forall i$, e restrição (i);
- Teste 3: $N_i = 30$, $\forall i$, $N_{ui} = 10$, $\forall i$, e restrições (i) e (ii);

Durante os testes as referências futuras não são conhecidas. Inicialmente $y_{r1} = 0,5$, $y_{r1} = 0,3$ e $y_{r1} = 0,1$. Em $t = 200$ min há uma mudança de referência para 0,4 na saída y_1, e em $t = 320$ min aplica-se uma perturbação na entrada u_1 de amplitude 0,05. Além disso, ruídos brancos de média zero e amplitudes máximas de $0,01$ são adicionadas às saídas. Erros de modelagem não são considerados nos testes.

Os gráficos na Fig. (38) mostram as saídas da planta e os sinais de controle aplicados durante o Teste 1. Nota-se que as amplitudes das entradas respeitam a restrição imposta, além disso, o controlador consegue fazer com que todas as saídas sigam as respectivas referências dentro de 80 min, um tempo consideravelmente menor do que em malha aberta. Uma outra característica importante é o desacoplamento realizado pelo algoritmo MPC, que pode ser visto nas respostas das saídas 2 e 3 que, após a mudança de referência da saída 1 em $t = 200$ min, praticamente não se alteram.

Na Tab. (7) são mostrados os tempos de otimização, tempo de outras tarefas (leitura de dados, multiplicação de matrizes, obtenção da resposta livre, etc.), o tempo total e também as quantidades relacionadas às variáveis de decisão (soma dos horizontes de controle) e às restrições do problema quadrático. Como já discutido anteriormente, não é possível saber *a priori* o tempo para resolver o problema

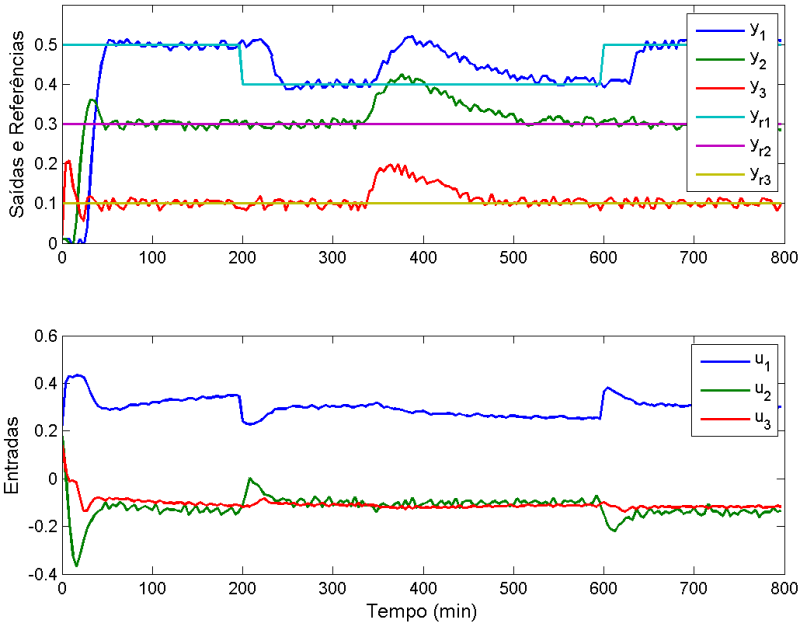


Figura 38 – Resultado do Teste 1 do exemplo 1.

Tabela 7 – Resultados dos testes do experimento 1. Os tempos, dados em segundos, encontram-se na forma $\bar{x}/\sigma$, onde $\bar{x}$ é o valor médio e σ o desvio padrão.

Teste	1	2	3
Variáveis de decisão	45	45	30
Quantidade de restrições	180	90	120
Tempo Otimização (s)	19,8/1,22	11,0/1,0	7,42/0,50
Tempo outras operações (s)	0,26/0,17	0,27/0,18	0,23/0,29
Tempo total (s)	20,1/1,24	11,3/1,0	7,65/0,60

quadrático MPC com o algoritmo de otimização utilizado. Além disso, devido à execução de tarefas secundárias existentes no SO do sistema embarcado, os tempos para completar as operações variam durante o experimento. Assim, devido à essa natureza não determinística, faz-se uso de uma abordagem estatística para analisar os dados temporais. Considera-se que os tempos seguem uma distribuição gaussiana com

média $\bar{x}$ e desvio padrão σ.

Pelos resultados dados, tem-se que o tempo necessário para calcular e enviar a ação de controle para a planta não passará de $23,82$ segundos com uma certeza estatística de 99 % ($\bar{x}+3\sigma$), um tempo muito menor que o período de amostragem da planta (4 minutos). Assim, o protótipo poderia ser utilizado para controlar esse processo.

A partir dos resultados também é possível visualizar a correlação entre a quantidade de variáveis de decisão, a quantidade de restrições e o tempo de otimização. Nota-se que, no teste 2, reduzindo a quantidade de restrições pela metade, o tempo de otimização cai 44 % e, no teste 3, reduzindo a quantidade de variáveis de decisão, o tempo cai ainda mais. Assim, caso o protótipo não conseguisse calcular a ação de controle dentro do período de amostragem, seria possível melhorar os tempos de otimização ajustando estes dois parâmetros. Porém, deve-se considerar que estes ajustes poderão prejudicar a resposta do sistema.

6.1.3 Experimento 2: Compressor

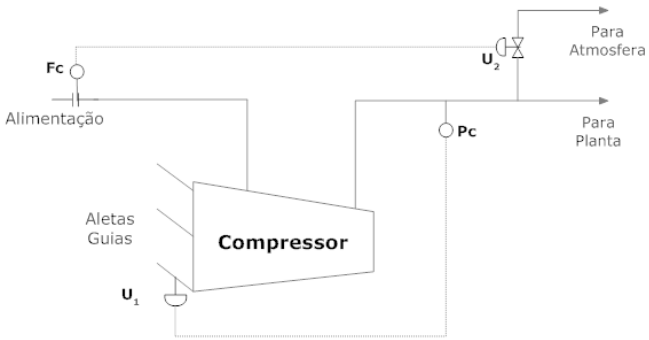


Figura 39 – Esquema do Compressor do exemplo 2.

O segundo experimento se trata do controle de um compressor de ar de grande escala encontrado frequentemente na indústria [5, p. 194]. A pressão de saída é controlada através da manipulação das aletas guias do compressor (esquema na Fig. (39)). Uma válvula de escape é instalada para prevenir picos de pressão. Quando esta válvula está fechada, o compressor é um sistema SISO que pode ser controlado com técnicas de controle simples. Quando a válvula abre, o compressor é um sistema MIMO com duas entradas e duas saídas. As variáveis

manipuladas são o ângulo das aletas guias (u_1) e a posição da válvula (u_2), e as variáveis controladas são a pressão (y_1) e a vazão de ar (y_2). O modelo do processo é dado pela equação a seguir, onde os tempos são dados em segundos:

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{0{,}1133e^{-0{,}715s}}{1+4{,}48s+1{,}783s^2} & \frac{0{,}9222}{1+2{,}071s} \\ \\ \frac{0{,}3378e^{-0{,}299s}}{1+1{,}09s+0{,}361s^2} & \frac{-0{,}321e^{-0{,}94}}{1+2{,}463s+0{,}104s^2} \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix} \quad (6.3)$$

Considerando um período de amostragem de $T_s = 0,05$ s, obtém-se o seguinte modelo discreto:

$$\begin{bmatrix} \frac{10^{-4}(0{,}7619z^{-1}+0{,}7307z^{-2})z^{-14}}{1-1{,}8806z^{-1}+0{,}8819z^{-2}} & \frac{0{,}022z^{-1}}{1-0{,}9761z^{-1}} \\ \\ \frac{10^{-2}(0{,}1112z^{-1}+0{,}1057z^{-2})z^{-6}}{1-1{,}8534z^{-1}+0{,}8598z^{-2}} & \frac{10^{-2}(-0{,}2692z^{-1}-0{,}1821z^{-2})z^{-19}}{1-1{,}2919z^{-1}+0{,}306z^{-2}} \end{bmatrix}. \quad (6.4)$$

Para esta planta, três testes foram feitos onde se adotou o algoritmo GPC. Em todos eles os pesos são constantes: $\boldsymbol{Q_y} = diag(\boldsymbol{I}, \boldsymbol{I})$ e $\boldsymbol{Q_u} = diag(0,8\boldsymbol{I}; 0,8\boldsymbol{I})$. Foram utilizadas restrições no valor absoluto da ação de controle e no seu incremento: (i) as entradas podem variar dentro do intervalo $[-2,75; 2,75]$ e (ii) valor máximo do incremento de controle de ± 1 por período de amostragem. As restrições utilizadas e os horizontes sofrem variações:

- Teste 1: $N_1 = 22$, $N_2 = 17$, $N_{ui} = 3$, $\forall i$, e restrição (i);
- Teste 2: $N_1 = 22$, $N_2 = 17$, $N_{ui} = 3$, $\forall i$, e restrições (i) e (ii);
- Teste 3: $N_1 = 22$, $N_2 = 17$, $N_{ui} = 6$, $\forall i$, e restrição (i);

Durante os testes há quatro mudanças de referência, em $t = 5$ s $y_{r1} = 1$, em $t = 15$ s $y_{r2} = 0,7$, em $t = 35$ s $y_{r1} = 0$ e, finalmente, em $t = 45$ s $y_{r2} = 0$. Assim como no experimento anterior, considera-se que as referências futuras não são conhecidas e também são adicionados ruídos brancos de média zero e amplitude máxima 0,001 nas saídas. Não são considerados erros de modelagem nos testes.

Na Fig. (40) é mostrado o resultado da simulação com este processo. Nota-se que, assim como no exemplo passado, o sistema embarcado foi capaz de gerar ações de controle dentro das restrições impostas.

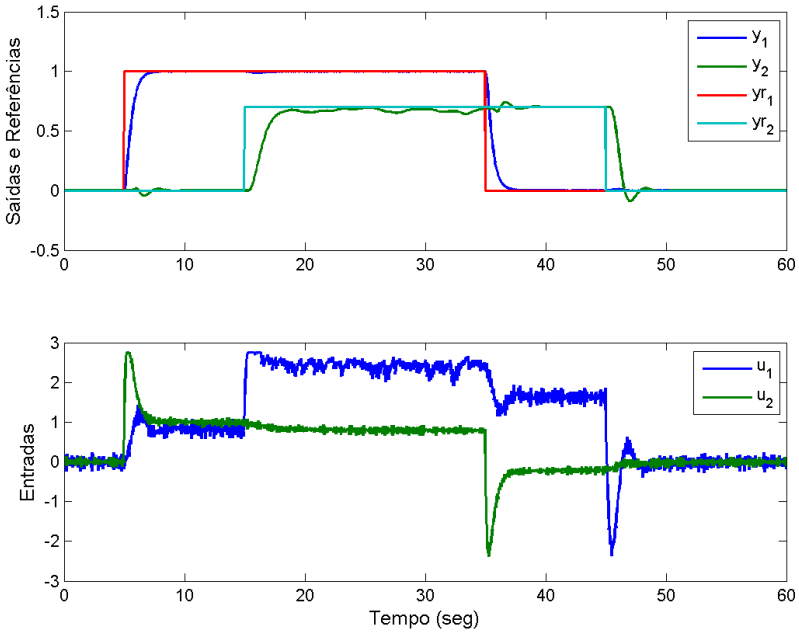


Figura 40 – Resultado do Teste 1 do exemplo 2.

Tabela 8 – Resultados dos testes experimento 2. Os tempos, dados em segundos, encontram-se na forma $\bar{x}/\sigma$, onde $\bar{x}$ é o valor médio e σ o desvio padrão.

Teste	1	2	3
Variáveis de decisão	6	6	12
Quantidade de restrições	12	36	24
Tempo Otimização (s)	0,48/0,12	0,63/0,14	1,04/0,27
Tempo outras operações (s)	0,09/0,16	0,09/0,15	0,10/0,14
Tempo total (s)	0,57/0,24	0,72/0,23	1,14/0,31

Assim como no exemplo anterior, pelos resultados apresentados na Tab. (8), aumentando os horizontes de controle e as restrições, aumenta-se o tempo de otimização. Mas, para esta planta, mesmo com um número baixo destes parâmetros, o protótipo não poderia ser utilizado no controle deste processo pois, mesmo no melhor caso (teste 1), levaria-se 1,29 segundos (com 99 % de certeza) para calcular e enviar

o sinal de controle à planta, um tempo cerca de 26 vezes maior do que o período de amostragem. Assim, verifica-se que, com o *hardware* atual, não é possível o controle de processos com taxas de amostragem rápidas.

6.1.4 Análise dos Resultados

Com os resultados obtidos nestes dois experimentos verificou-se que a versão atual do protótipo é capaz de ser utilizada em processos com taxas de amostragem na ordem de minutos, como visto no experimento 1. Como o público alvo do produto criado é justamente médios e pequenos processos de dinâmicas lentas, o protótipo conseguiu atingir o objetivo estipulado. É possível até mesmo controlar processos com períodos de amostragem na faixa de dezenas de segundos, contanto que os horizontes de controle e a quantidade de restrições utilizadas sejam ajustadas adequadamente, mas deve-se levar em conta que, caso estes ajustes sejam feitos apenas levando em conta o tempo de amostragem, a resposta do sistema em malha fechada poderá sofrer deterioração.

6.2 EXPERIMENTO 3: COLUNA DE DESTILAÇÃO DE ETANOL

O segundo experimento também se trata de uma simulação HIL, mas há a substituição do simulador Python pelo programa Aspen HYSYS que faz simulações realistas de processos químicos. Os resultados obtidos com este experimento permitiram a elaboração do artigo [65], que foi aceito para publicação no *European Control Conference* (ECC-2013), a ser realizado em julho de 2013 em Zürich.

Na indústria brasileira de etanol, o processo de destilação é comumente encontrado na configuração ilustrada na Figura (41), onde é apresentada a presença de dois estágios: destilação e retificação. Os produtos gerados nestes estágios são os etanóis de segunda e hidratado. Os sub-produtos são a vinhaça, flegmaça e o óleo fúsel.

A unidade de destilação tem um papel significativo na indústria de etanol já que este estágio da produção consome a maior parte da energia utilizada no processo de produção. Apesar disso, infelizmente, é muito comum encontrar uma operação inadequada dos sistemas de controle que, na maioria dos casos, leva a um consumo de energia maior e à degradação da qualidade da produção [65, 66]. Isso se deve à dificuldade de entendimento do processo por operadores e engenheiros, o que leva a diferentes conclusões sobre a forma que o sistema deve

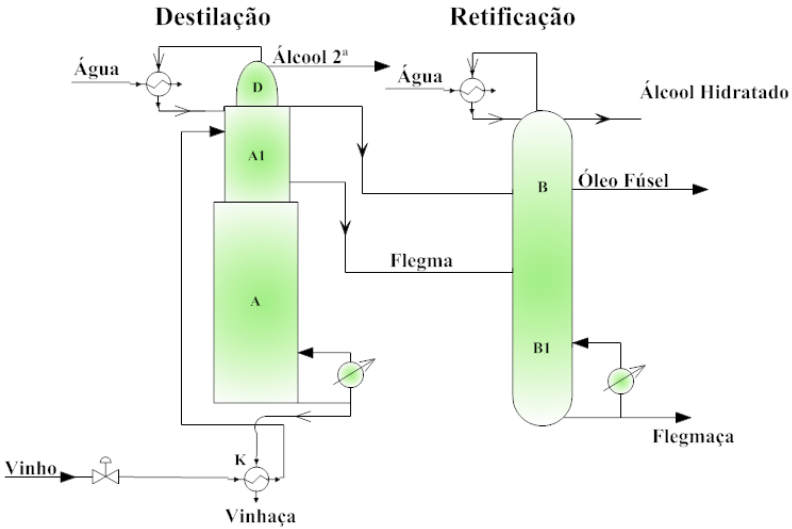


Figura 41 – Esquema da coluna de destilação de etanol usada no experimento 2.

ser controlado. Esta controvérsia parte das inúmeras possibilidades de configuração da coluna e das complexas interações das suas malhas, que, por sua vez, também dependem das propriedades das colunas de destilação. Assim, não há uma solução geral do ponto de vista de controle, apenas princípios físico-químicos que, propriamente aplicados, permitem a operação bem sucedida destes sistemas [67].

Para estudar este problema, uma simulação completa de uma destilaria de etanol foi desenvolvida como parte da tese de doutorado de Marcus Americano da Costa [68]. Esta simulação foi feita utilizando o *software* Aspen HYSYS, que é um sistema de modelagem líder de mercado, utilizado pelas grandes empresas da área de petróleo e gás [69], que é capaz de criar simulações muito realistas, a ponto de ser utilizado para projeto de novas colunas de destilação e para estudos de colunas existentes. Assim, com o uso deste simulador, os dados de simulação se tornam muito mais verídicos, ajudando a promover o produto desenvolvido. Na Fig. (42) é mostrada a tela do programa HYSYS onde foi criada a simulação deste experimento.

Apesar do grande número de estratégias alternativas para melhorar a operação deste tipo de processo, o uso do controle preditivo está crescendo cada vez mais e pode ser um solução viável para o au-

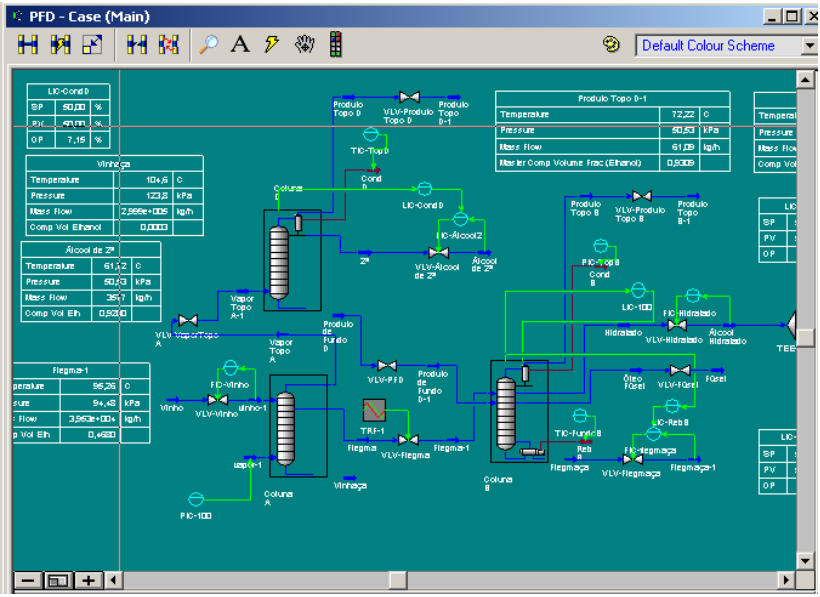


Figura 42 – Tela do programa Aspen HYSYS com o arquivo de simulação utilizado no experimento 2.

mento da eficiência energética na produção de etanol [66]. Por este fator, e considerando que o *software* de simulação utiliza um modelo bastante realista de uma unidade de destilação, este experimento foi desenvolvido para mostrar de maneira mais adequada a capacidade do sistema embarcado criado.

Assim, foram estabelecidos os seguintes objetivos para este experimento:

- Desenvolver um experimento para verificar o desempenho do protótipo criado de forma que este fique sujeito a condições de operação próximas das de um processo real;

- Análise do comportamento do sistema: tempo de computação, erros, etc.;

6.2.1 Descrição do Processo

O processo de destilação é baseado na diferença de volatibilidade dos seus constituintes, caracterizado por uma dupla mudança de estado físico. Inicialmente, uma substância no estado líquido é aquecida até atingir a temperatura de ebulição, transformando-se em vapor. Esta fase entra em contato com a fase líquida, ocorrendo uma transferência de massa do líquido para o vapor e deste para aquele. O líquido e vapor contêm, em geral, os mesmos componentes, mas em quantidades relativas diferentes. O líquido está em seu ponto de bolha[1] e o vapor em equilíbrio, no seu ponto de orvalho[2]. Existe uma transferência simultânea de massa do líquido pela vaporização e do vapor pela condensação. O efeito final é o aumento do componente mais volátil no vapor e do componente menos volátil no líquido. Posteriormente, o vapor é resfriado até que toda a massa retorne ao estado líquido, assim obtendo uma substância líquida onde o componente em maior abundância é diferente do componente principal da substância original [66, 70].

Nestes tipos de processos, as principais variáveis controladas são: concentração dos produtos de topo e fundo das colunas (as composições são frequentemente inferidas a partir de temperaturas que representam os pontos de ebulição nas pressões de operação), níveis de líquido na base da coluna e no tanque acumulador, e temperatura e pressão da coluna. É importante frisar que todas as vazões devem ser medidas, incluindo a alimentação, e devem-se aplicar controladores secundários nos fluxos manipuláveis de vapor [71]. As variáveis manipuladas são as vazões dos produtos, dos fluidos de aquecimento e resfriamento; e refluxo. Este último proporciona a existência da fase líquida no último estágio da coluna, retirando calor do topo. Normalmente, não é possível manipular a alimentação porque esta é um produto oriundo de uma outra coluna [65, 66].

Neste caso particular, o vinho, que é produto de uma etapa anterior da produção de etanol (processo de fermentação), é aquecido e alimenta a coluna A1 (ver Fig. (41)). A composição do vinho é de, aproximadamente, 90% água, etanol (7 a 10 ^{o}GL) e outras substâncias encontradas em menores quantidades tais como glicerina, ácido acético, metanol, etc. O vinho começa a ferver e a perder a maior parte dos seus componentes mais voláteis que sobem para a coluna D. Da base da coluna A1, o vinho passa para a coluna A e desce, perdendo álcool até que o fundo da coluna A é atingido, gerando a vinhaça. No topo

[1] Temperatura na qual a vaporização se inicia.
[2] Temperatura na qual a condensação se inicia

da coluna A retira-se flegma na forma de vapor que é levada à coluna B1 para retificação. No topo da coluna D, o vapor é condensado e um dos produtos é obtido, o álcool de segunda a uma concentração de 92 ^{o}GL. Enquanto isso, o produto na base desta coluna, que é rico em etanol, é levado para a coluna B para retificação. A base da coluna B é alimentada com flegma (a uma concentração de 50 ^{o}GL), que subirá pela coluna, ficando cada vez mais enriquecida com etanol, até que atinja o topo onde será condensada e retirada na forma de álcool hidratado (concentração de 96 ^{o}GL). Nos níveis mais baixos da coluna B1, a solução formará flegmaça, um sub-produto muito pobre em etanol.

6.2.2 Controle da Unidade de Destilação

Como este processo funciona de forma contínua estabilizado em um determinado ponto de operação, o principal problema de controle está na rejeição de perturbações. Assim, os objetivos de controle estabelecidos foram:

- manter a concentração de álcool de segunda (y_1) em 92 ^{o}GL;
- manter a concentração de álcool hidratado (y_2) em 96 ^{o}GL;
- manter a concentração de vinhaça (y_3) menor do que $0,04$ ^{o}GL.

Para que isto ocorra, o controlador MPC funcionará em cascata com controladores PID locais manipulando as seguintes variáveis: pressão de fundo da coluna A (u_1), temperatura de topo da coluna D (u_2) e o nível do tanque do condensador na coluna B (u_3). As perturbações mensuráveis consideradas neste problema são: (i) vazão de alimentação de vinho (d_1), (ii) vazão de flegma (d_2), e (iii) vazão de saída do álcool hidratado (d_3). As duas últimas são ajustadas de acordo com as diretivas de produção.

Apesar de as colunas terem dinâmicas não-lineares, o algoritmo MPC utilizado calcula a ação de controle a partir de modelos lineares. Desta forma, primeiro é necessário identificar os modelos linearizados do processo no ponto de operação desejado para depois ajustar o controlador. Os pontos de operação escolhidos foram, para as saídas, $\bar{y}_1 = 92$ ^{o}GL para o álcool de segunda, $\bar{y}_2 = 0,03$ ^{o}GL para a vinhaça, e $\bar{y}_3 = 96$ ^{o}GL para o álcool hidratado. Para as entradas, $\bar{u}_1 = 119,1$ kPa para a pressão de fundo, $\bar{u}_2 = 72,38$ ^{o}C para a temperatura de topo e $\bar{u}_3 = 50\%$ do nível máximo do tanque do condensador da coluna B. Finalmente, para as perturbações, $\bar{d}_1 = 295000$ kg/h para a alimentação

de vinho, $\bar{d}_2 = 39550$ kg/h para a vazão de flegma e $\bar{d}_3 = 18130\ kg/h$ para a vazão de álcool hidratado.

A identificação do modelo do processo foi feita aplicando-se sequências de degraus unitários nas entradas e perturbações. Após a aplicação destes degraus, os dados coletados foram analisados com o auxílio da biblioteca de identificação de sistemas (*System Identification Toolbox*) do programa Matlab, obtendo-se, assim, os modelos entrada-saída e perturbação-saída da planta.

Os modelos entrada-saída obtidos foram normalizados de forma a permitir um ajuste mais fácil das ponderações do controlador preditivo. As saídas do processo, que referenciam concentrações, já estão normalizadas (variam de 0 a 1 ^{o}GL), restando apenas normalizar as entradas e perturbações. Os valores máximos permitidos para as entradas são: $u_{1,max} = 150$ kPa, $u_{2,max} = 140\ ^o$C e $u_{3,max} = 100$ %. Assim, para normalizar as entradas, basta dividir cada uma delas por, respectivamente, 150 kPa, 140 ^{o}C e 100 %. Para as perturbações, o valor máximo considerado para todas elas é $d_{i,max} = 3,6$ kg/h, $\forall i$.

Após a identificação, os modelos normalizados entrada-saída, $P_u(s)$ na Eq. (6.5), e perturbação-saída, $P_d(s)$ na Eq. (6.6), foram obtidos no domínio de Laplace. A equação é $\boldsymbol{\Delta y}(s) = P_u \boldsymbol{\Delta u}(s) + P_d \boldsymbol{\Delta d}(s)$ onde $\boldsymbol{\Delta y} = [\Delta y_1, \Delta y_2, \Delta y_3]^T$, $\boldsymbol{\Delta u} = [\Delta u_1, \Delta u_2, \Delta u_3]^T$ e $\boldsymbol{\Delta d} = [\Delta d_1, \Delta d_2, \Delta d_3]^T$.

$$P_u = \begin{bmatrix} \frac{-2,6443}{1971,2s+1} & \frac{6,9829}{1706,3s+1} & 0 \\ \frac{-4,4526 \cdot 10^{-2}}{1610s+1} & \frac{1,6649 \cdot 10^{-2}}{2392,8s+1} & 0 \\ 0 & 0 & \frac{-9,674 \cdot 10^{-3}(508,86s+1)}{(222,35s+1)(15,074s+1)} \end{bmatrix} \tag{6.5}$$

$$P_d = \begin{bmatrix} \frac{0,121}{1445,7s+1} & \frac{0,201}{1880,93s+1} & 0 \\ \frac{7,28 \cdot 10^{-3}}{815,89s+1} & \frac{-20,66 \cdot 10^{-3}}{204,26s+1} & 0 \\ 0 & \frac{0,205}{s} & \frac{0,05}{s} \end{bmatrix} \tag{6.6}$$

6.2.3 Parâmetros do Experimento

As condições deste experimento foram as seguintes:

a) tempo de simulação de 500 minutos;

b) o processo de destilação está em regime permanente no ponto de operação escolhido;

c) a alimentação de vinho será alterada para simular uma variação de $\pm 1,7\%$ em sua produção pelo processo anterior (fermentação): $t = 2$ min aumento de 1,7 %, $t = 121$ min volta ao valor nominal, $t = 242$ min redução de 1,7 e em $t = 362$ min volta novamente ao valor nominal;

d) alterações na vazão de flegma através da variação da abertura da válvula que controla esta variável para compensar parcialmente a variação na alimentação de vinho: $t = 5$ min aumento da abertura em 2,5 %, $t = 125$ min volta ao valor nominal, $t = 245$ min diminuição da abertura em 2,5 %, $t = 365$ min volta ao valor nominal;

e) alterações na vazão de álcool hidratado devido à variação da alimentação de vinho: $t = 15$ min aumento de 6,5 %, $t = 135$ volta ao valor nominal, $t = 255$ min diminuição de 6,5 %, $t = 375$ min volta ao valor nominal;

f) o controlador MPC sendo executado no sistema embarcado terá que manter as concentrações dos produtos nos valores especificados na seção 6.2.2.

Depois de um ajuste inicial do controlador e de diversas simulações com o modelo linearizado, os parâmetros do controlador GPC linear foram escolhidos. O período de amostragem é de $T_s = 60$ s. Os horizontes de predição são $N_1 = N_2 = 60$ para, respectivamente, o álcool de segunda e a vinhaça, e $N_3 = 30$ para o álcool hidratado. Os pesos das ações de controle são $\boldsymbol{Q_u} = diag(2\boldsymbol{I}; 4\boldsymbol{I}; 0,1\boldsymbol{I})$ e $\boldsymbol{Q_y} = diag(0,01\boldsymbol{I}; 0,01\boldsymbol{I}; 8\boldsymbol{I})$ são os pesos das saídas. As restrições, quando consideradas, são: (i) valores máximos e mínimos para as variáveis manipuladas, $\boldsymbol{U_{max}} = [121,1\,\text{kPa}; 75,38\,^o\text{C}; 90\,\%]$, e $\boldsymbol{U_{min}} = [117,1\,\text{kPa};\ 69,38\,^o\text{C}; 10\,\%]$; (ii) máxima variação das entradas em um período de amostragem $\Delta u_1 = \pm 1$ kPa/min, $\Delta u_2 = \pm 1\ ^o$C/min, e $\Delta u_3 = \pm 10\,\%$/min.

Os horizontes de controle e restrições variam durante os testes da seguinte forma:

- Teste 1: $N_{u_1} = N_{u_2} = 15$, $N_{u_3} = 5$ e restrições (i) e (ii);
- Teste 2: mesmos parâmetros do Teste 1, mas somente restrição (i);
- Teste 3: $N_{u_1} = N_{u_2} = 20$, $N_{u_3} = 10$ e restrições (i) e (ii);

Estes parâmetros devem fazer com que perturbações do tipo degrau sejam rejeitadas em menos de 20 minutos, de acordo com as simulações com o modelo linearizado.

6.2.4 Resultados

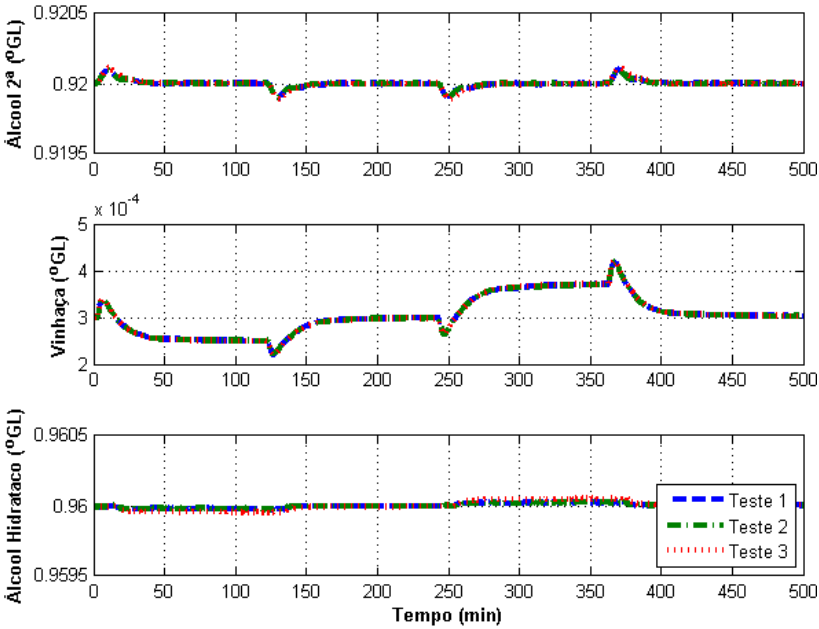


Figura 43 – Valores das saídas nos diferentes testes para o controle da coluna de destilação de etanol.

Na Fig. 43, é possível ver os valores das saídas do processo de destilação nos diferentes testes e, na Fig. (44), os valores das variáveis manipuladas. Em todos os testes os controladores conseguem rejeitar as perturbações em menos de 20 minutos para o caso do álcool de segunda, que é a variável mais afetada. Quanto à vinhaça, os controladores conseguiram manter sua concentração abaixo de 0,04 ºGL, como foi especificado. Já a concentração de álcool hidratado sofre variações muito pequenas em todos os casos.

Os dados mais interessantes, no entanto, são os tempos de com-

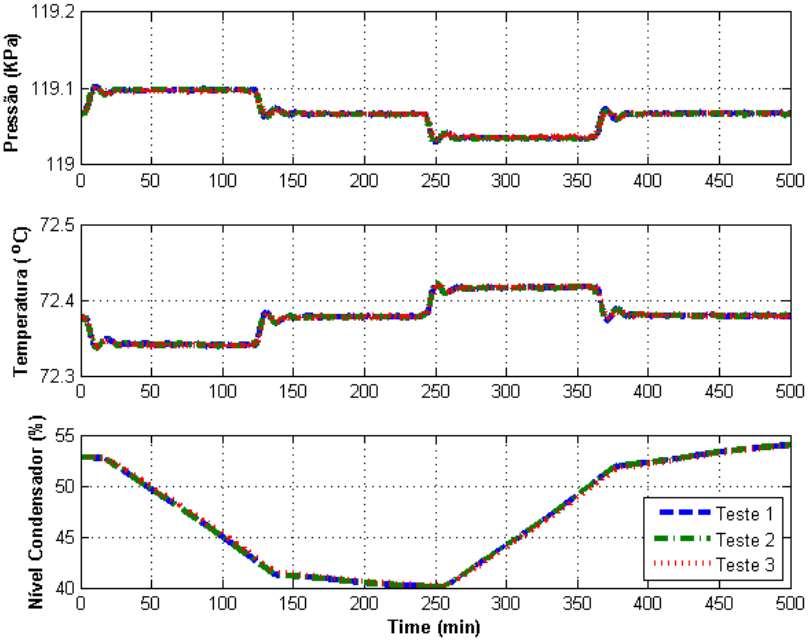


Figura 44 – Valores das entradas nos diferentes testes para o controle da coluna de destilação de etanol.

putação do sistema embarcado para calcular o sinal de controle do processo. Um dos principais problemas de implementação dos algoritmos MPC é a necessidade de resolver um problema de otimização em tempo real. Isto impede a aplicação do MPC em muitos processos, seja porque a tecnologia computacional requerida para resolver a otimização dentro de período de amostragem seja cara demais ou mesmo indisponível, seja porque o programa implementado que resolve a otimização não obedeça a critérios de certificação de *software* , especificamente em aplicações onde a segurança é um aspecto crítico [72]. Foi importante registrar o tempo de computação de cada tarefa sendo executada pelo sistema embarcado durante a simulação HIL para ter uma estimação da capacidade computacional do sistema embarcado e estipular um limite em termos de tempo de amostragem e restrições. Isto indicará com que tipo de processos o sistema atual poderá trabalhar.

Na Tab. (9) são mostrados os tempos referentes ao diferentes testes deste experimento. Nota-se que, mesmo no pior caso (Teste 3),

Tabela 9 – Resultados dos testes experimento 3. Os tempos, dados em segundos, encontram-se na forma $\bar{x}/\sigma$, onde $\bar{x}$ é o valor médio e σ o desvio padrão.

Teste	1	2	3
Variáveis de Decisão	35	35	50
Quantidade de Restrições	140	70	200
Tempo Otimização (s)	$14,25/2,60$	$5,87/0,65$	$34,0/5,47$
Tempo outras operações (s)	$0,28/0,25$	$0,26/0,13$	$0,32/0,14$
Tempo total (s)	$14,52/2,60$	$6,12/0,67$	$34,31/5,46$

onde o número de restrições e de variáveis de decisão são maiores, o tempo necessário para calcular a ação de controle é de 50,7 segundos com 99 % de certeza. No entanto, este é um tempo muito próximo do tempo de amostragem. Desta forma, é sugerível aplicar os parâmetros do Teste 1, ou seja, reduzir os horizontes de controle. Veja que isto não comprometerá a resposta do sistema, como visto na Fig. (43), e permitirá que o tempo total seja reduzido para 22,32 segundos com 99% de certeza. Neste caso então, seria viável a utilização do protótipo para o controle deste processo.

6.3 EXPERIMENTO 4: INTEGRAÇÃO COM PROCESSO REAL

Para a validação final do protótipo, tinha-se como objetivo o controle de um processo real com o sistema embarcado. Para tanto, utilizou-se a Planta Didática III da SMAR [73], localizada no Departamento de Automação e Sistemas da UFSC.

O objetivo da Planta Didática SMAR é demonstrar didaticamente a operação das diversas malhas de controle presentes na planta utilizando os mesmos equipamentos e ferramentas de configuração desenvolvidos para aplicação em controle industrial [74]. O processo, cuja esquematização simplificada pode ser vista na Fig. (45), é constituído de três tanques em cascata que formam um circuito fechado. O Tanque de Alimentação fornece a água que, pelo uso de uma bomba centrífuga, alimenta o Tanque 1. A vazão de alimentação deste tanque é controlada pela abertura da válvula V_1. Quando o Tanque 1 está cheio, a água excedente é transportada por uma tubulação para o Tanque 2 que possui formato cônico, diferentemente dos outros tanques que possuem formato cilíndrico. A água que chega ao Tanque 2 escoa de volta para

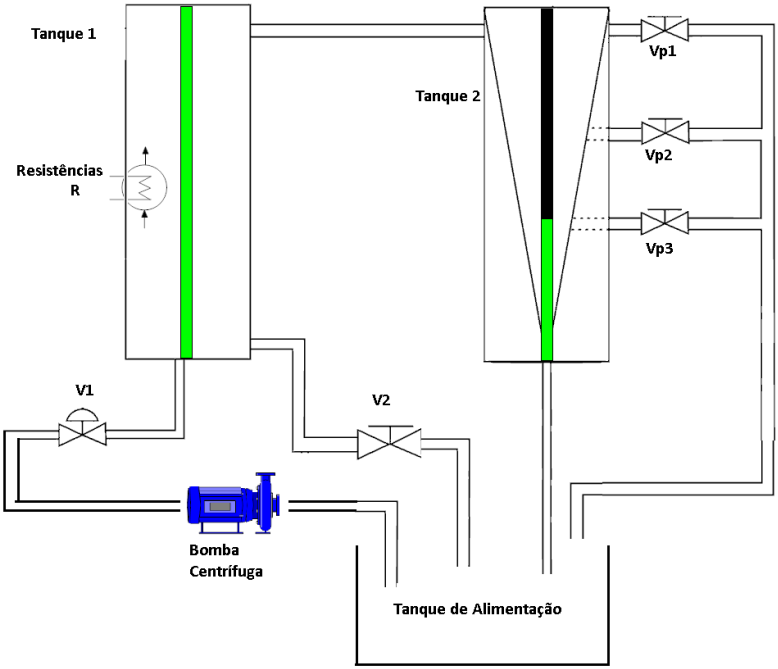


Figura 45 – Esquema da Planta Didática III da SMAR.

o Tanque de Alimentação através de uma tubulação em seu fundo, e também através de saídas auxiliares que podem ser abertas através das válvulas manuais V_{p1}, V_{p2} e V_{p3}, de forma a introduzir perturbações no sistema. Também é possível controlar a temperatura da água que circula pelo processo ao acionar as resistências elétricas de imersão, representadas por R, presentes dentro do Tanque 1. Desta forma, decidiu-se realizar um experimento onde se deseja controlar o nível do Tanque 2 e a temperatura da água circulante medida no Tanque 1 através a abertura da válvula V_1 e do acionamento das resistências R.

A instrumentação desta planta didática é fundamentada na tecnologia FOUNDATION Fieldbus. São utilizados instrumentos inteligentes, capazes de executar distribuidamente e de forma dedicada todo o controle do processo. Entre os instrumentos, encontram-se válvulas de controle FY302 [18], rotâmetros para detecção de fluxo no tubulação, sensores de nível, resistências de imersão para aquecimento da água, bomba centrífuga, entre outros. Além disso, existe um dispositivo de

ligação (LAS), modelo DFI302, responsável pelo gerenciamento da rede FF e um CLP (LC700) que coordena os equipamentos que não suportam a rede FF e também realiza o intertravamento da planta.

Como o gerenciamento da rede FF é feito pelo dispositivo de ligação DFI302, é a partir dele que será feita a integração com o protótipo desenvolvido através da rede MODBUS. O DFI302 já possui uma entrada serial RS232, então, para instalar o protótipo, foi necessário apenas a conexão de um cabo entre os equipamentos. Na Fig. (46) é mostrado o esquema da integração entre o processo existente e o protótipo.

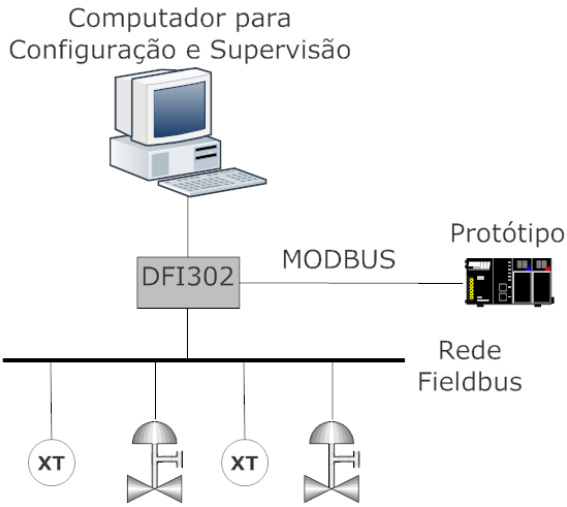


Figura 46 – Esquema dos dispositivos da planta didática e sua integração com o protótipo.

Instalado o sistema embarcado, primeiramente foi feita a configuração da comunicação no dispositivo DFI302. A configuração deste dispositivo e de todos os equipamentos FF é feita através do *software* Smar SYSCON. Através deste programa, a configuração dos equipamentos e dos laços de controle na rede FF se resume a adicionar blocos que representam diferentes funções (controlador PID, entrada analógica, saída analógica, etc.), e fazer as ligações apropriadas entre os blocos adicionados. Na tela mostrada na Fig. (47) pode ser visto um exemplo onde há a ligação entre blocos MODBUS, blocos de entradas e saídas analógicas e blocos de dois controladores PID no programa SYSCON.

Portanto, para configurar a comunicação MODBUS, foi preciso adicionar os blocos Configuração MODBUS (MBCF), que configura como serão transmitidos os dados através da rede, e Controle MODBUS Mestre (MBCM), que é o bloco responsável por ler e escrever dados através da rede MODBUS. Na Fig. (48) é mostrada a tela do SYSCON depois de adicionados e configurados os blocos MODBUS.

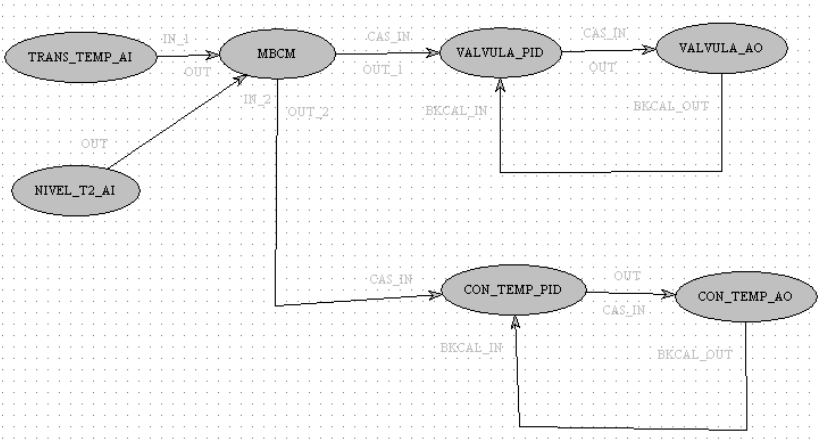


Figura 47 – Tela de adição de blocos do programa SYSCON.

Após a configuração adequada dos parâmetros destes blocos, feita com o auxílio do manual fornecido pela Smar [75], verificou-se se a integração entre rede FF e protótipo estava funcionando corretamente. Para isso, através do programa SYSCON, foram feitas operações de leitura e escrita na memória do sistema embarcado através da rede industrial, o que ocorreu sem nenhum problema.

6.3.1 Controle da Planta Didática

Com o protótipo integrado ao processo, restou realizar os procedimentos de ajuste do controlador MPC. Primeiramente é preciso identificar o modelo do processo. Para isso, com a planta no ponto de operação escolhido, aplicou-se sequências de degraus nas entradas do processo: abertura da válvula (u_1) e potência (u_2). Depois de coletados os dados de identificação, chegou-se ao seguinte modelo linearizado, onde os tempos são dados em segundos:

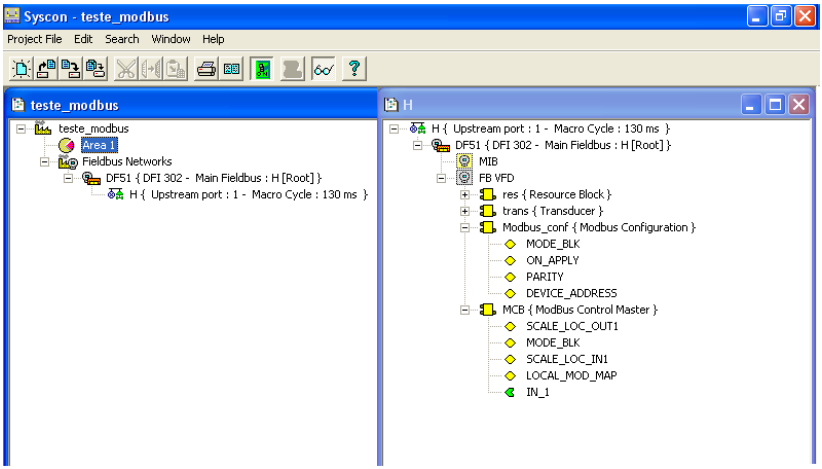


Figura 48 – Tela principal do programa SYSCON após a adição dos blocos MODBUS.

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{1{,}44}{59{,}62s+1} & 0 \\ \frac{-1{,}8\cdot 10^{-4}}{s} & \frac{5{,}62\cdot 10^{-4}}{s} \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix}. \tag{6.7}$$

onde y_1 e y_2 são, respectivamente, o nível do Tanque 2 e a temperatura do Tanque 1. Todas as variáveis estão normalizadas e variam de 0 a 100, e todas as unidades estão em porcentagem, com exceção da temperatura que é medida em graus centígrados mas, por variar entre 0 e 100 ^{o}C, esta se encontra normalizada em relação às outras variáveis. O ponto de operação escolhido foi: $\bar{y}_1 = 40\%$, $\bar{y}_2 = 40\,^o$C, $\bar{u}_1 = 60\%$ e $\bar{u}_2 = 0\%$. É importante explicar que a dinâmica da temperatura em relação às entradas é integradora devido ao fato de o processo ser um circuito fechado, isto é, a água circula continuamente entre os três tanques. Assim, o que faz a temperatura da água diminuir é a troca térmica com o ambiente, que é pequena em relação ao fornecimento de calor pelas resistências, o que faz a temperatura subir continuamente caso as resistências sejam ligadas.

Como o objetivo principal deste experimento é mostrar a integração do protótipo com um processo real e não as vantagens de algoritmos preditivos, optou-se por um ajuste simples do controlador

de forma a apenas garantir seguimento de referência e rejeição de perturbações com tempo de assentamento próximo do tempo de malha aberta. Através dos resultados de simulações, adotou-se o tempo de amostragem de $T_s = 15$ s e chegou-se aos seguintes parâmetros de ajuste para o algoritmo GPC: $N_1 = 10$, $N_2 = 20$, $N_{u_1} = N_{u_2} = 5$ e $\boldsymbol{Q_y} = \boldsymbol{Q_u} = diag(\boldsymbol{I}, \boldsymbol{I})$. Foram utilizadas restrições apenas nos valores absolutos dos sinais de controle: $[U_{1,min}, U_{1,max}] = [55\,\%, 75\,\%]$ e $[U_{2,min}, U_{2,max}] = [0\,\%, 30\,\%]$. Este ajuste, pelas simulações com o modelo nominal, fará com que o tempo de assentamento do sistema seja de aproximadamente 3 minutos para o nível e de 8,5 minutos para a temperatura.

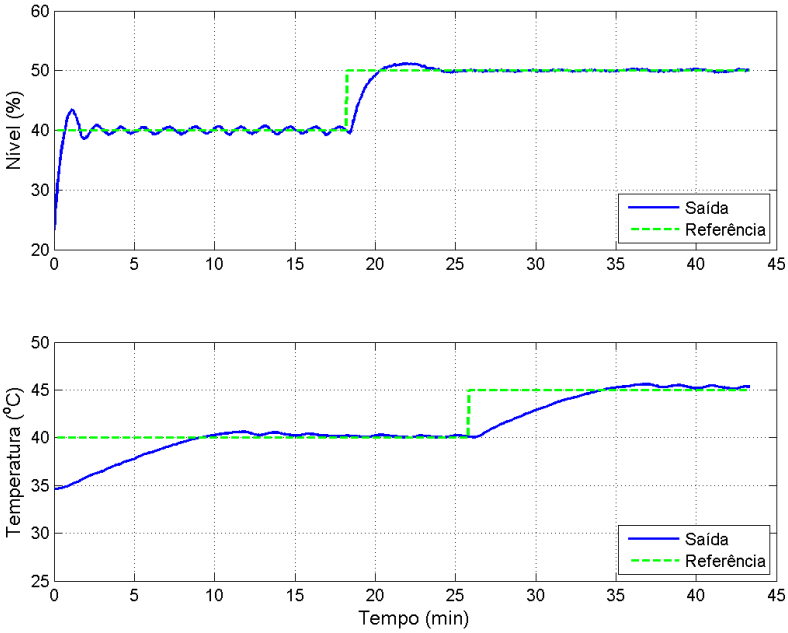


Figura 49 – Saídas da planta SMAR no experimento 3.

6.3.2 Resultados

Nas figuras (49) e (50) são mostrados, respectivamente, os gráficos das saídas e entradas do processo durante a realização deste ex-

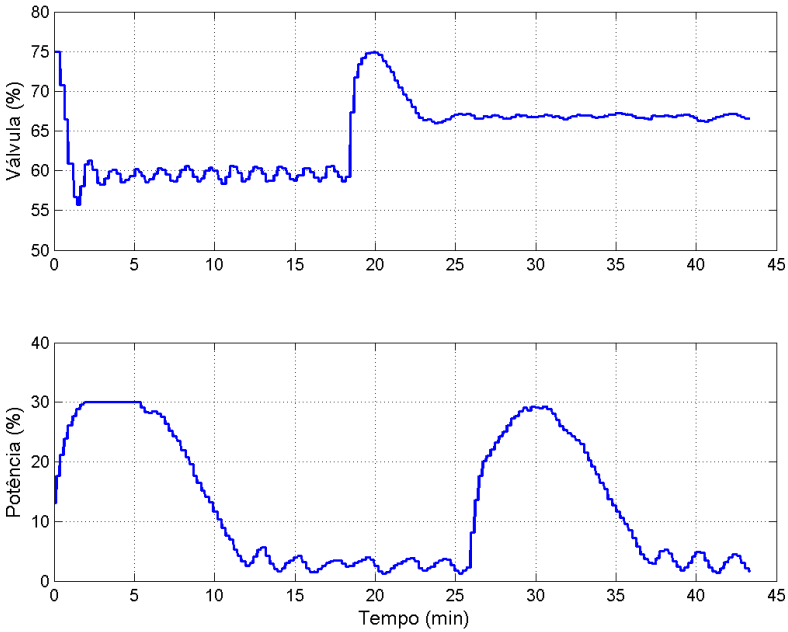


Figura 50 – Entradas da planta SMAR no experimento 3.

perimento. O controle através do sistema embarcado é iniciado com o processo fora do ponto de operação, com o nível em 20 % e a temperatura em 35 ^{o}C. O controlador projetado leva cerca de 2 minutos para levar a planta ao nível do ponto de operação e aproximadamente 10 minutos no caso da temperatura, o que está de acordo com os resultados obtidos com a simulação do modelo do processo considerando que existem erros de modelagem. Em $t = 18,2$ min muda-se a referência de nível para 50 % e em $t = 25,8$ min a referência de temperatura passa a ser 45 ^{o}C. Percebe-se que o controlador projetado é capaz de estabilizar o sistema e de seguir as referências dadas sem violar as restrições estipuladas. Nota-se, no entanto, oscilações no sinal de controle de potência. Isto é causado pela resolução do acionamento dos resistores, que não consegue aplicar um valor de potência muito pequeno. Assim, o controlador é forçado a aplicar ciclicamente um valor maior e menor fazendo a temperatura da água sofrer pequenas oscilações.

Ao final do experimento, foram recuperados os dados temporais relacionados ao processamento feito pelo sistema embarcado, que são

sintetizados na Tab. (10). O tempo requerido pelo protótipo para gerar a ação de controle é 2,76 segundos com certeza de 99 %, cerca de cinco vezes menor que o tempo de amostragem, o que, junto com os resultados mostrados nas figuras (49) e (50), mostra que o protótipo é capaz de controlar esse sistema adequadamente.

Tabela 10 – Resultados temporais do experimento 4. Os tempos, dados em segundos, encontram-se na forma $\bar{x}/\sigma$, onde $\bar{x}$ é o valor médio e σ o desvio padrão.

Teste	Planta SMAR
Variáveis de decisão	10
Quantidade de restrições	20
Tempo Otimização	1,45/0,31
Tempo outras operações	0,11/0,17
Tempo total	1,56/0,4

6.4 CONCLUSÕES

Durante este capítulo, foram mostrados alguns dos experimentos realizados para validar o protótipo criado. Inicialmente foi feita uma avaliação da capacidade computacional do protótipo através de experimentos HIL com exemplos apresentados em livros sobre controle preditivo. Depois foi feita um experimento mais realista utilizando um simulador profissional de processos químicos. Por último, para mostrar de forma concreta que o protótipo desenvolvido funciona, realizou-se um experimento com uma planta real que possui dispositivos FOUNDATION Fieldbus.

Com os resultados apresentados neste capítulo, mostra-se que o protótipo produzido é capaz de atingir o objetivo especificado para o projeto: controle de plantas de médio e pequeno porte com dinâmicas lentas.

O próximo capítulo apresentará as conclusões finais deste projeto e as propostas de trabalhos futuros.

7 CONCLUSÃO

Este trabalho mostrou os resultados obtidos no desenvolvimento de um controlador MPC baseado em um sistema embarcado de baixo custo visando processos industriais de pequeno e médio porte de dinâmicas lentas onde o uso de algoritmos preditivos, se bem ajustados, permitem melhorar a qualidade da produção. Controladores MPC não são adotados nestes tipos de processos porque os investimentos em técnicas avançadas de controle não são, atualmente, economicamente viáveis. O produto desenvolvido tenta solucionar este problema oferecendo um equipamento de baixo custo facilmente integrável com processos existentes, e que suporta diferentes protocolos de comunicação, o que fará com que sua instalação seja mais fácil e barata pois utilizará a infra-estrutura existente da planta.

No capítulo introdutório deste documento foram apresentadas as características que o produto desenvolvido deveria ter. Estas características são listadas novamente a seguir, com comentários sobre os resultados obtidos:

- Fácil integração com processos industriais existentes através do suporte a diferentes redes industriais, por exemplo, Fieldbus, MODBUS, PROFIBUS-PA: isto foi obtido com a implementação do protocolo MODBUS, pois é um protocolo aberto e livre de licenças que é suportado por praticamente todas as redes industriais mais novas;

- Configuração rápida dos parâmetros do controlador: isto é feito através de um arquivo XML produzido através da Interface de Usuário;

- Suporte a diferentes algoritmos MPC, permitindo que o usuário escolha a melhor opção para um determinado processo: foram implementados os algoritmos GPC, DTCGPC e SSMPC.

Os resultados dos experimentos reforçam as características listadas. Foi mostrado que o protótipo é capaz de trabalhar com plantas de pequeno e médio porte com dinâmicas lentas (Seção 6.1 e 6.3) e que a versão atual já é capaz de ser integrada à infra-estrutura de processos existentes sem muito trabalho (Seção 6.3).

7.1 TRABALHOS FUTUROS

Apesar dos bons resultados obtidos, algumas funcionalidades precisam ser adicionadas e outras, já existentes, melhoradas. Assim, sugere-se que, na continuação deste projeto, os seguintes pontos sejam considerados:

- Algoritmos MPC:
  a) Suporte ao uso de funções objetivos diferentes, que permitam um ajuste melhor do controlador MPC;
  b) Adição de algoritmos baseados em respostas ao degrau, tal como o DMC;
  c) Adição de algoritmos MPC não-lineares como o PNMPC [39];
  d) Adição de um observador de estados no algoritmo SSMPC quando não é possível obter a leitura direta de todos os estados;
  e) Adição de técnicas de controle adaptativo;
- Otimização:
  a) Implementação de um algoritmo de otimização próprio, de forma a ter controle maior sobre esta tarefa. Deverá ser adicionado a este algoritmo a capacidade de detecção e tratamento de problemas quadráticos infactíveis e também se a solução será obtida dentro do período de amostragem, como discutido na Seção 4.6.2;
  b) Adição de técnicas de redução de restrições, de forma a reduzir o tempo de otimização;
  c) Análise para verificar se a implementação de uma biblioteca de otimização em C ou C++ reduziria de forma relevante o tempo de otimização;
- *Software* do sistema embarcado:
  a) Caso seja implementado uma biblioteca de otimização determinística, torna-se possível a utilização de sistemas operacionais de tempo real, pois todas as outras operações do *software* são determinísticas;

b) Ajustes ou alteração da metodologia de desenvolvimento de *software* adotada para acomodar os aspectos de tempo real inerentes à aplicação de controle de processos, podendo fazer uso de *frameworks* de projetos de sistemas de tempo real como o SIMOO-RT [76];

c) Avaliação da existência de condições de *deadlock* e outros testes de desempenho como WCET (*Worst Case Execution Time*), que mede o pior tempo de execução de um algoritmo, e SIL (*Safety Integrity Level*), que proporciona uma medida probabilística de quando o dispositivo sofrerá uma falha perigosa;

- Interface de Usuário:

a) Melhoramentos da interface para torná-la mais amigável;

b) Adição de técnicas de identificação de sistemas para auxiliar o usuário na obtenção do modelo do processo;

Para tornar o protótipo criado pronto para ser utilizado em um processo real, deve-se levar em conta principalmente a questão de factibilidade e obtenção da ação de controle dentro do período de amostragem, o uso de um SO de tempo real para garantir determinismo para a aplicação e a questão dos testes de desempenho de *software* mencionados.

REFERÊNCIAS

1 WIKIMIDIA-COMMONS. *Centrifugal governor*. 2011. Disponível em: <http://goo.gl/io2fR>. Acesso em: 29/01/2013.

2 SEGOVIA, V.; THEORIN, A. *History of Control - PLC and DCS*. 2012. Disponível em: <http://goo.gl/mDe58>. Acesso em: 29/01/2013.

3 PASETTI, G. *Instrumentação, Controle e Supervisão de uma Coluna de Destilação Piloto Utilizando Tecnologia Foundation Fieldbus*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina - UFSC, 2005.

4 PINHEIRO, B. C. *Sistema de Controle Tempo Real Embarcado para Automação de Manobra de Estacionamento*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina - UFSC, 2009.

5 CAMACHO, E.; BORDONS, C. *Model Predictive Control*. [S.l.]: Springer, 1999. (Advanced Textbooks in Control and Signal Processing).

6 CLARKE, D.; MOHTADI, C. Properties of generalized predictive control. *Automatica*, v. 25, n. 6, p. 859 – 875, 1989.

7 LEVINE, W. *The Control Handbook, Second Edition (Three Volume Set)*. [S.l.]: CRC PressINC, 2010. (Electrical Engineering Handbook).

8 EMERSON Process Products and Services. 2012. Disponível em: <http://emersonprocess.com>. Acesso em: 31 jan. 2013.

9 MILLS, A.; WILLS, A.; NINNESS, B. Nonlinear model predictive control of an inverted pendulum. In: *ACC09*. [S.l.: s.n.], 2009.

10 WILLS, A.; MILLS, A.; NINNESS, B. FPGA Implementation of an Interior-Point Solution for Linear Model Predictive. In: *18th IFAC World Congress 2011, Milan, Italy*. [s.n.], 2011. Disponível em: <http://goo.gl/80H0E>.

11 LING, K.; YUE, S.; MACIEJOWSKI, J. A FPGA implementation of model predictive control. In: *American Control Conference, 2006*. [S.l.: s.n.].

12 VALENCIA-PALOMO, G.; ROSSITER, J. Programmable logic controller implementation of an auto-tuned predictive control based on minimal plant information. {*ISA*} *Transactions*, v. 50, n. 1, p. 92 – 100, 2011.

13 ATTA Tecnologia de Trafégo. 2012. Disponível em: <http://www.attatrafego.com.br>. Acesso em: 20 nov. 2012.

14 VARGAS, L. A. *Sistemas de Controle*. 2000. Disponível em: <http://goo.gl/LRP41>. Acesso em: 28 jan. 2013.

15 FERNÁNDEZ-CARA, E.; ZUAZUA, E. Control Theory: History, Mathematical Achievements and Perspectives. *Bol. Soc. Esp. Mat. Apl.*, 2009.

16 BENNETT, S. *A History of Control Engineering, 1800-1930*. [S.l.]: Peregrinus, 1986. (IEE Control Engineering Series).

17 BERGE, J. *Fieldbuses for Process Control: Engineering, Operation and Maintenance*. [S.l.]: ISA, The Instrumentation, Systems, and Automation Society, 2002.

18 SMAR. *FY302 - Posicionador de Válvula Fieldbus Foundation (Produto SMAR)*. 2010. Disponível em: <http://goo.gl/rgD94>. Acesso em: 28/01/2013.

19 SMAR. *Fieldbus Tutorial - A Foundation Fieldbus Technology Overview*. 2010. Disponível em: <http://goo.gl/rgD94>. Acesso em: 28/01/2013.

20 IDC. *Practical Fieldbus, Devicenet & Ethernet for Industry*. IDC Technologies, 2002. Disponível em: <http://goo.gl/8Szxy>.

21 FIELDBUSFOUNDATION. *Foundation Technical Specifications Index*. 2012. Disponível em: <http://goo.gl/aiXPC>. Acesso em: 28/01/2013.

22 FIELDBUSFOUNDATION. *Foundation Fieldbus Testing and Registration Services*. 2012. Disponível em: <http://goo.gl/ZWdNL>. Acesso em: 28/01/2013.

23 MODBUS. *Modbus Application Specification V1.1b*. 2006. Disponível em: <http://goo.gl/i30rf>. Acesso em: 29/01/2013.

24 EMBEDDEDARM. *Technologic Systems - Embedded Arm.* 2012. Disponível em: <http://www.embeddedarm.com/>. Acesso em: 30 jan. 2013.

25 SILBERSCHATZ, A.; GAGNE, G.; GALVIN, P. *Operating System Concepts.* [S.l.]: Wiley, 2011.

26 TANENBAUM, A.; AUSTIN, T. *Structured Computer Organization.* [S.l.]: Prentice Hall, 2000.

27 BARBALACE, A. et al. Performance Comparison of VxWorks, Linux, RTAI, and Xenomai in a Hard Real-Time Application. *Nuclear Science, IEEE Transactions on*, v. 55, n. 1, p. 435 –439, fev. 2008.

28 HUGHES, J. *Real World Instrumentation with Python: Automated Data Acquisition and Control Systems.* [S.l.]: O'Reilly Media, 2010. (Oreilly Series).

29 NUMPY - biblioteca matemática para Python. 2012. Disponível em: <http://www.numpy.org/>. Acesso em: 30 jan. 2013.

30 ANDERSEN, M.; DAHL, J.; VANDENBERGHE, L. *Python Software for Convex Optimization.* 2012. Disponível em: <http://abel.ee.ucla.edu/cvxopt/>. Acesso em: 30 jan. 2013.

31 MODBUSTK - Modbus Test Kit. 2012. Disponível em: <http://code.google.com/p/modbus-tk/>. Acesso em: 30 jan. 2013.

32 SCIPY - A beginners guide to uusing Python for performance computing. 2011. Disponível em: <http://scipy.org/PerformancePython>. Acesso em: 30 jan. 2013.

33 VANDENBERGHE, L. *The CVXOPT linear and quadratic cone program solvers.* 2010. Disponível em: <http://goo.gl/eFlVu>. Acesso em: 30 jan. 2013.

34 RICHALET, J. et al. Algorithm control for industrial processes. In: *Proc. 4th IFAC Symp. on Identification and System Parameter Estimation.* [S.l.: s.n.], 1976.

35 CUTLER, C.; RAMAKER, B. L. Dynamic matrix control-a computer control algorithm. *AIChE National Meeting*, 1979.

36 KEYSER, R. D.; CUAWENBERGHE, A. Extended prediction self adaptive control. In: *IFAC Simp. on Ident. and Syst. Parameter Estimation.* [S.l.: s.n.], 1985. p. 1317–1322.

37 YDSTIE, B. Extended horizon adaptive control. In: *9th IFAC World Congress*. [S.l.: s.n.], 1984.

38 QIN, S.; BADGWELL, T. A. A survey of industrial model predictive control technology . *Control Engineering Practice*, v. 11, n. 7, p. 733 – 764, 2003.

39 PLUCENIO, A. *Desenvolvimento de Técnicas de Controle Não Linear para Elevação de Fluídos Multifásicos*. Tese (Doutorado) — Universidade Federal de Santa Catarina, 2010.

40 CLARKE, D. W.; MOHTADI, C.; TUFFS, P. S. Generalized predictive control - Part I. The basic algorithm. *Automatica*, p. 137–148, 1987.

41 NORMEY-RICO, J.; CAMACHO, E. *Control of Dead-Time Processes*. [S.l.]: Springer, 2007. (Advanced Textbooks in Control and Signal Processing).

42 SKOGESTAD, S.; POSTLETHWAITE, I. *Multivariable feedback control: analysis and design*. [S.l.]: John Wiley, 2005.

43 ROSSITER, J. *Model-Based Predictive Control: A Practical Approach*. [S.l.]: CRC Press, 2003. (CRC Press Control Series).

44 CLARKE, D.; YOON, T. *Advances in Model-Based Predictive Control*. [S.l.]: Oxford University Press, 1994. (Oxford Science Publications, v. 2).

45 NORMEY-RICO, J.; CAMACHO, E. Robustness effects of a prefilter in a smith predictor-based generalised predictive controller. *Control Theory and Applications, IEE Proceedings* -, v. 146, n. 2, p. 179 –185, mar 1999.

46 NORMEY-RICO, J.; CAMACHO, E. Multivariable generalised predictive controller based on the smith predictor. *Control Theory and Applications, IEE Proceedings* -, v. 147, n. 5, p. 538 –546, sep 2000.

47 TSANG, T. T. C.; CLARKE, D. Generalised predictive control with input constraints. *Control Theory and Applications, IEE Proceedings D*, v. 135, n. 6, p. 451–460, 1988.

48 SOTOMAYOR, O. A.; ODLOAK, D.; MORO, L. F. Closed-loop model re-identification of processes under MPC with zone control. *Control Engineering Practice*, v. 17, n. 5, p. 551 – 563, 2009.

49 HENRIKSSON, D.; ÅKESSON, J. *Flexible Implementation of Model Predictive Control Using Sub-optimal Solutions*. [s.n.]. (Institutionen för reglerteknik, Lunds tekniska högskola). Disponível em: <http://goo.gl/HG9o1>.

50 WAZLAWICK, R. *Análise e Projeto de Sistemas de Informação Orientados a Objetos*. [S.l.]: Elsevier, 2004.

51 LARMAN, C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design, and the Unified Process*. [S.l.]: Prentice Hall PTR, 2002. (Safari electronic books).

52 SCOTT, K. *The unified process explained*. [S.l.]: Addison-Wesley, 2002.

53 PEREIRA, C. E.; CARRO, L. Distributed real-time embedded systems: Recent advances, future trends and their impact on manufacturing plant control. *Annual Reviews in Control*, Elsevier, v. 31, n. 1, p. 81–92, 2007.

54 DEITEL, H.; DEITEL, P. *C+: How to Program 7th Edition*. [S.l.]: Pearson Prentice Hall, 2010. (How to program series).

55 JOHNSON, R. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. [S.l.]: Addison-Wesley, 1995.

56 W3C - Extensible Markup Language (XML) 1.0 (Fifth Edition). 2013. Disponível em: <http://www.w3.org/TR/REC-xml/>. Acesso em: 11 fev. 2013.

57 PYTHON. *Minimal DOM implementation*. 2013. Disponível em: <http://docs.python.org/2/library/xml.dom.minidom.html>. Acesso em: 11 fev. 2013.

58 JONES, C.; DRAKE, F. *Python and XML*. [S.l.]: O'Reilly Media, 2002. (Oreilly Series).

59 LIMA, D. M. *XML Schema utilizado neste projeto*. 2012. Disponível em: <http://goo.gl/L19zW>.

60 NABI, S. et al. An overview of hardware-in-the-loop testing systems at Visteon. 2004. Disponível em: <http://goo.gl/42ytN>. Acesso em: 18/02/2013.

61 MODELING and stability analysis of a simulation-stimulation interface for hardware-in-the-loop applications. *Simulation Modelling Practice and Theory*, v. 15, n. 6, p. 734 – 746, 2007. Disponível em: <http://goo.gl/xld4s>.

62 ABULQUERQUE, A. de. A técnica de hardware-in-the-loop no auxílio de projetos mecatrônicos. *Mecatrônica Atual*, Agosto 2007. Disponível em: <http://goo.gl/I99VZ>.

63 BULLOCK, D. et al. Hardware-in-the-loop simulation. *Transportation Research Part C: Emerging Technologies*, v. 12, n. 1, p. 73 – 89, 2004. Disponível em: <http://goo.gl/ATfsA>.

64 PRETT, D. M.; MORARI, M. Shell Process Control Workshop, Butterworths, Stoneham, MA, 1987. *AIChE Journal*, American Institute of Chemical Engineers, v. 35, n. 5, p. 876–876, 1989.

65 LIMA, D. M.; COSTA, M. V. A. da; NORMEY-RICO, J. E. A flexible low cost embedded system for model predictive control of industrial processes. Aceito para publicação no European Control Conference (ECC2013).

66 COSTA, M. V. A. da; CRUZ, D. M.; NORMEY-RICO, J. E. Modelagem, simulação e controle de uma unidade de destilação em uma usina produtora de etanol. *XIX Congresso Brasileiro de Automática, Campina Grande-PB, 2012.*

67 SKOGESTAD, S. Dynamics and Control of Distillation Columns - A Critical Survey. *Modeling, Identification and Control*, Norwegian Society of Automatic Control, v. 18, n. 3, p. 177–217, 1997. Disponível em: <http://goo.gl/8BV8x>.

68 COSTA, M. V. A. da. *Modelagem, Controle e Otimização de Processos da Indústria do Etanol: uma aplicação em Energia Solar.* Tese (Doutorado) — Universidade Federal de Santa Catarina, 2013.

69 ASPENTECH. *Aspen HYSYS®*. 2013. Disponível em: <http://www.aspentech.com/hysys/>. Acesso em: 11 fev. 2013.

70 FOUST, A. et al. *Princípios das operações unitárias.* [S.l.]: Guanabara Dois, 1982.

71 SHINSKEY, F. *Process control systems: application, design, and tuning.* [S.l.]: McGraw-Hill, 1996. (Chemical engineering books).

72 ALESSIO, A.; BEMPORAD, A. A survey on explicit model predictive control. In: MAGNI, L.; RAIMONDO, D.; ALLGöWER, F. (Ed.). *Nonlinear Model Predictive Control*. [S.l.]: Springer Berlin Heidelberg, 2009, (Lecture Notes in Control and Information Sciences, v. 384). p. 345–369.

73 SMAR. *Planta Didática SMAR - Tanques em Cascata*. 2012. Disponível em: <http://goo.gl/AcpZ6>. Acesso em: 01/03/2013.

74 SMAR. *SMAR Didactic - Planta Didática III*. [S.l.], 2002.

75 SMAR. *Manual de Instruções dos Blocos Funcionais*. [S.l.], 2007.

76 BECKER, L.; PEREIRA, C. SIMOO-RT - An object-oriented framework for the development of real-time industrial automation systems. *Robotics and Automation, IEEE Transactions on*, v. 18, n. 4, p. 421–430, 2002.

APÊNDICE A – Documentos do Projeto de *Software*

A.1 TABELAS DE REQUISITOS

Tabela 11 – Tabela descrevendo o requisito funcional F1 - Atualizar novos dados.

F1 - Atualizar dados do processo
Descrição - os dados recebidos através da rede passa por um processo de conversão linear ($\boldsymbol{y} = \boldsymbol{Ax} + \boldsymbol{B}$) e, no caso do sensor, filtragem. Depois são armazenados adequadamente para uso no cálcula da ação de controle.
Restrições Não-Funcionais
NF1.1 - Possibilidade de ajuste dos coeficientes lineares de conversão $\boldsymbol{A}$ e $\boldsymbol{B}$.
NF1.2 - os dados são: referências, saídas, perturbações e sinais de controle.

Tabela 12 – Tabela descrevendo o requisito funcional F3 - Alterar Modo de Controle.

F3 - Alterar Modo de Controle
Descrição - permite a comutação entre modo automático e manual.
Restrições Não-Funcionais
NF1.1 - Permitir que o usuário troque a qualquer momento o modo de controle.
NF1.2 - o algoritmo MPC deve continuar a ser atualizado para, quando ocorrer uma troca de modo manual para automático, as variáveis internas já estejam atualizadas.

Tabela 13 – Tabela descrevendo o requisito funcional F4 - Transmissão de dados.

F4 - Transmissão de dados
Descrição - Receber e enviar dados através da rede industrial disponível.
Restrições Não-Funcionais
NF1.1 - Permitir a escolha entre redes MODBUS, ETHERNET, RS232, RS485.
NF1.2 - ser tolerante a falhas de comunicação.
NF1.3 - registrar quando os novos dados foram recebidos, de forma assim, se um certo dado não for atualizado durante um período determinado, é acionado um alarme devido a falta de atualização de dados.

Tabela 14 – Tabela descrevendo o requisito funcional F5 - Configuração MPC.

F5 - Configuração do MPC
Descrição - determinação dos parâmetros de configuração do MPC através da Interface de Usuário.
Restrições Não-Funcionais
NF1.1 - Permitir escolha dos algoritmos preditivo
NF1.2 - Permitir a escolha de quais restrições serão utilizadas
NF1.3 - A Interface de Usuário deve permitir a simulação do controlador projetado para verificar se o ajuste foi feito de forma adequada.

Tabela 15 – Tabela descrevendo o requisito funcional F6 - Configuração Rede Industrial.

F6 - Configuração do MPC
Descrição - Permitir a escolha do tipo de rede e a definição de seus parâmetros de configuração.

Tabela 16 – Tabela descrevendo o requisito funcional F7 - Verificação do Estado do Programa.

F7 - Verificação do Estado do Programa
Descrição - Permitir a verificação de erros e do estado atual do sistema embarcado através da Interface de Usuário.

A.2 CLASSES DOS PROJETO DE *SOFTWARE*

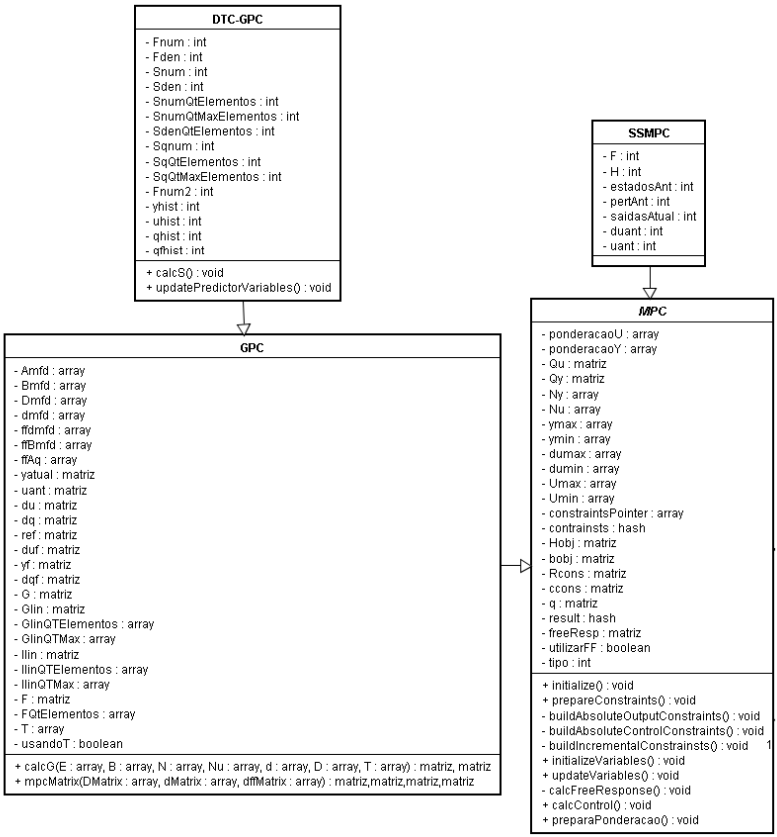


Figura 51 – Classes dos algoritmos de controle.

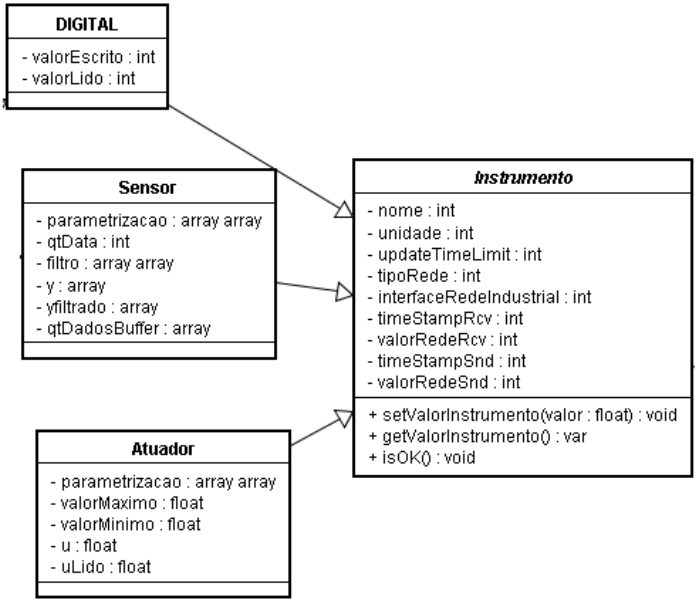


Figura 52 – Classes dos instrumentos.

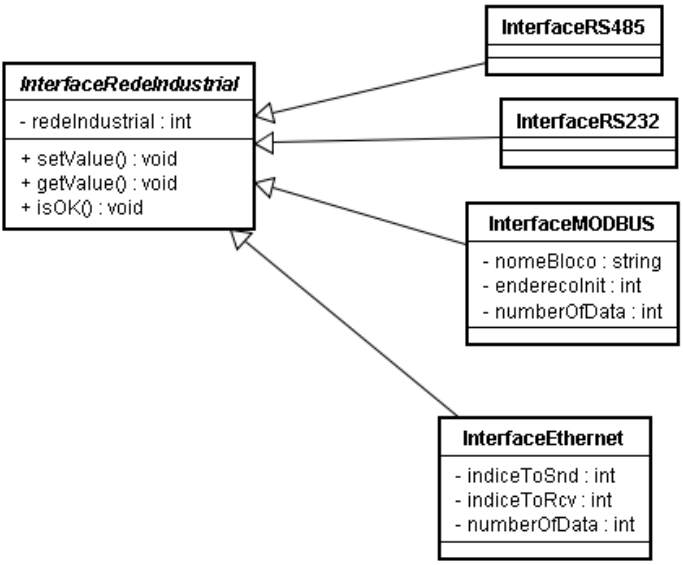


Figura 53 – Classes das interfaces de rede.

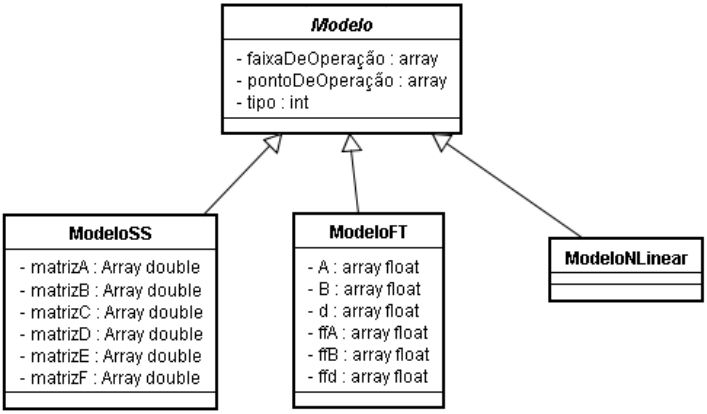


Figura 54 – Classes dos modelos do processo.

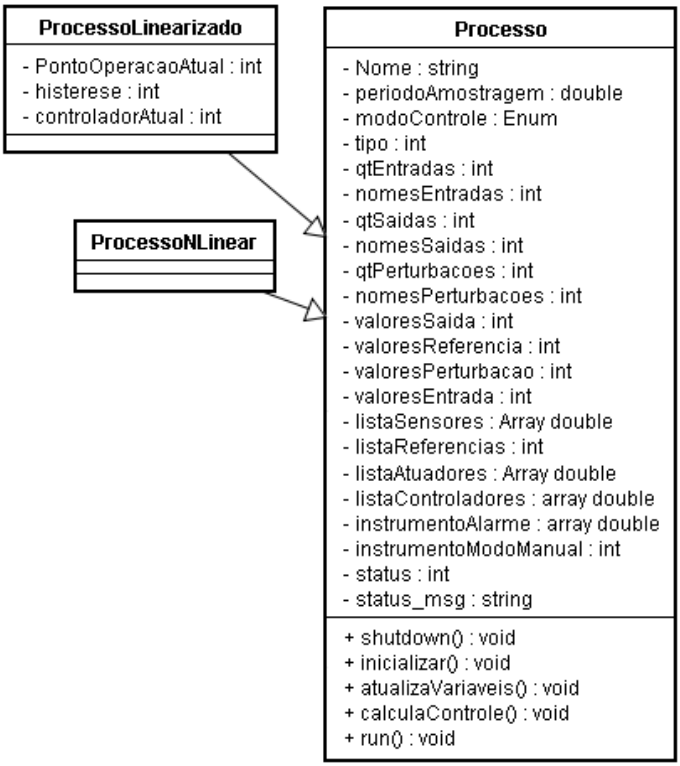


Figura 55 – Classes dos processos.

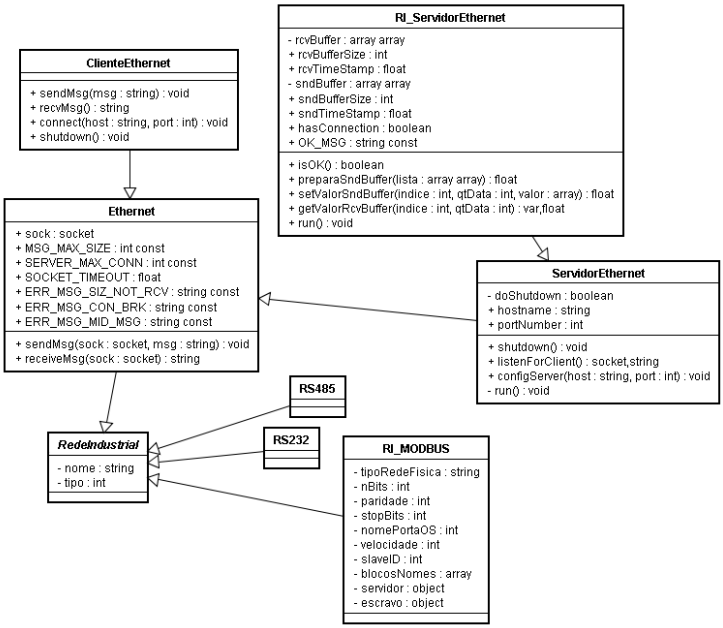


Figura 56 – Classe das redes industriais.